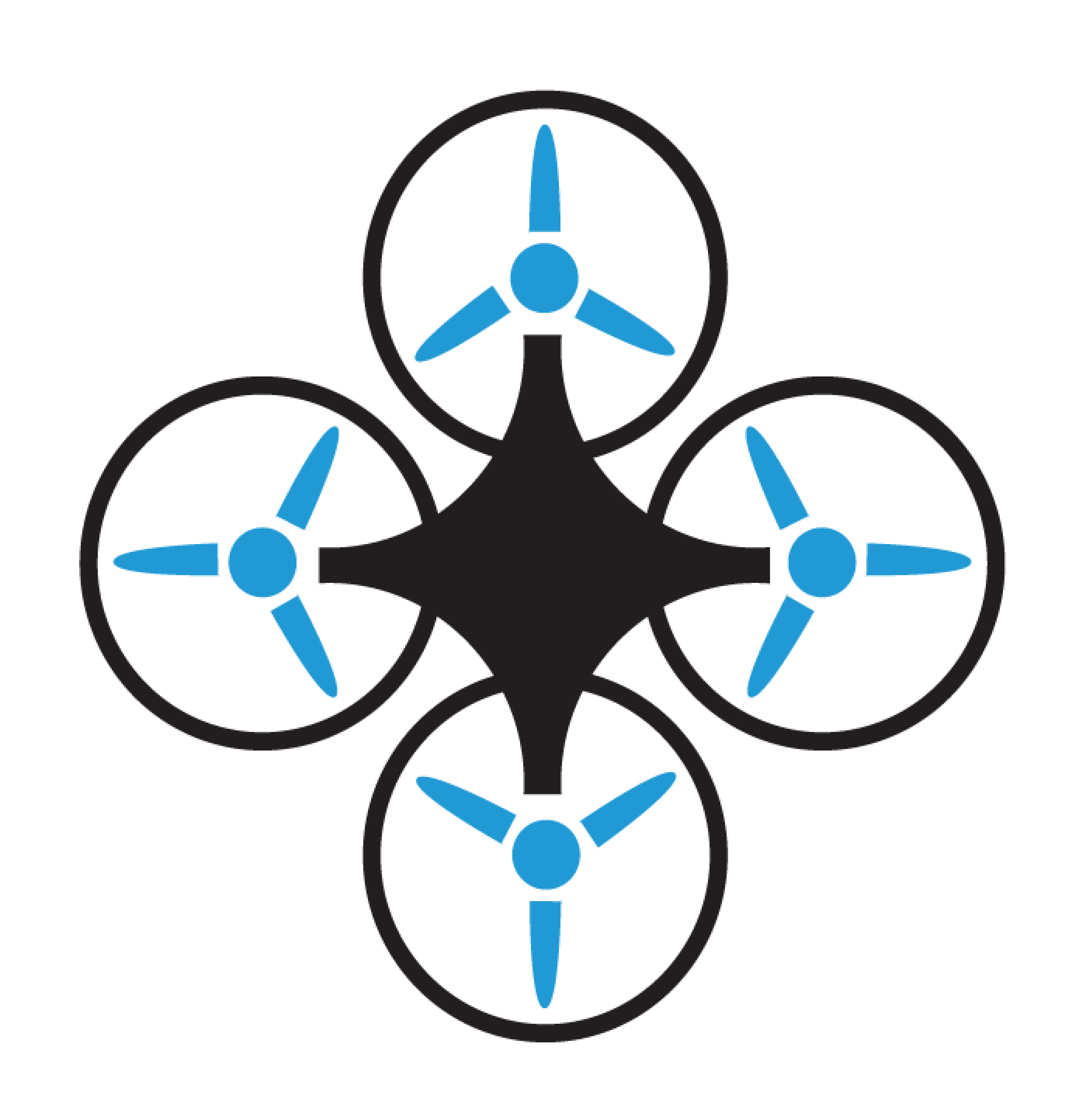
大学生电子设计竞赛案例汇编

四旋翼飞行器硬件框架自主设计

四旋翼飞行器软件系统自主设计与开发

四旋翼飞行器飞行控制程序自主开发与实践





Design and Implementation of the Quadrotors

四旋翼飞行器设计与实现

王 瑞　丁晓青◎编著

Wang Rui　Ding Xiaoqing

清華大学出版社

清华开发者书库

四旋翼飞行器设计与实现

王　瑞　丁晓青　编著

清华大学出版社
北　京

内 容 简 介

本书从四旋翼飞行器的历史和现状切入，通过阐述无人机的飞行原理、系统构成、通信协议等，向初学者介绍无人机入门的基础知识；从实际开发和应用入手，详细介绍了自主设计的四旋翼飞行器的硬件框架、常用硬件及其性能和用法、自主开发的软件系统设计以及飞行控制程序的开发与实践；另外，本书还展示了四旋翼飞行器的组装过程，提供了实现无人机平稳飞行的实用调试经验，列举了多项实际应用案例。本书致力于帮助读者从不会到会，自主搭建四旋翼飞行器、自主编写程序，并在自制的调试平台上分析数据和调节 PID 参数，列举了初次起飞时的常见问题及其解决方法，直到飞行器最终能平稳飞行。

本书适用于对四旋翼飞行器感兴趣的初学者入门学习，也可作为参加各种大学生无人机电子设计竞赛的大学生们的学习参考用书。

图书在版编目(CIP)数据

四旋翼飞行器设计与实现/王瑞，丁晓青编著. —北京：清华大学出版社，2018
(清华开发者书库)
ISBN 978-7-302-48964-1

Ⅰ. ①四…　Ⅱ. ①王…　②丁…　Ⅲ. ①旋翼机—设计　Ⅳ. ①V275

中国版本图书馆 CIP 数据核字(2017)第 292824 号

责任编辑：梁　颖　赵晓宁
封面设计：李召霞
责任校对：梁　毅
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：186mm×240mm　　**印　　张**：11.75　　**字　　数**：286 千字
版　　次：2018 年 7 月第 1 版　　**印　　次**：2018 年 7 月第 1 次印刷
印　　数：1～2500
定　　价：49.00 元

产品编号：063028-01

前言

PREFACE

1907年，世界上第一架四旋翼飞行器成功升空，拉开了四旋翼飞行器研究的大幕。四旋翼飞行器的研究，是多学科的碰撞和结合，涉及控制理论、电子与通信理论、图像处理技术及各种传感器技术，也与材料科学等学科密切相关。

由于四旋翼飞行器相较于固定翼飞行器具有机动性强、能源利用率高、结构设计巧妙和安全性高等特点，相较于地面机器人具有更高的自由度，因此近年来已成为一个研究热点，不仅大学等研究机构进行了广泛而深入的研究，以大疆为首的消费级四旋翼飞行器也取得了爆发式的发展。

四旋翼飞行器作为一个载体，在实现基本飞行功能的基础上，可以附加各种不同的功能，这也是四旋翼飞行器的最大魅力之一。近年来还出现了各类新型无人机，如自拍无人机、农业植保无人机和快递无人机等。各大研究机构开发的室内精确定位无人机则能实现更多精细化的工作，如利用无人机搭建垂直村落、无人机集群化协同飞行表演等。

实现四旋翼飞行器自主飞行功能的基础：①四旋翼飞行器实现自主平衡；②四旋翼飞行器在有遥控的情况下按照遥控者意愿飞行，且操作难度较低；③四旋翼飞行器在无遥控情况下利用附加传感器实现定点悬停和位移控制。

对于初学者来说，四旋翼飞行器基础飞行功能的实现有较高的难度，不仅要根据飞行原理装配四旋翼飞行器的硬件，在硬件层面保证飞行器能够安全稳定飞行，而且要进行四旋翼飞行器软件设计，尤其是较为困难的底层软件设计。此外，四旋翼飞行器参数的调节对开发者来说也是巨大的挑战。

本书深入浅出地全面介绍了四旋翼飞行器的各方面知识，让读者能轻松容易地进行学习。各章节分布如下：第1章介绍四旋翼飞行器的历史、现状和关键技术；第2章介绍四旋翼飞行器的飞行原理，包括姿态表示、姿态解算、平衡控制算法和滤波算法；第3章介绍四旋翼飞行器设计中常用的通信协议；第4章介绍四旋翼飞行器上常用的硬件设备；第5章介绍四旋翼飞行器软件系统设计；第6章介绍飞行控制模块PCB的设计及四

旋翼飞行器的组装经验和调试经验；第 7 章介绍几项四旋翼飞行器的实际应用案例。

本书的撰写得到了上海大学无人机协会成员徐晨畅、钱思杰、曹中正、黄潇畅、胡啸林等的大力协助，对他们在本书编写过程中所做的工作表示感谢。

由于编者的知识水平有限，书中疏漏之处在所难免，希望广大读者批评指正。

编　者

2018 年 3 月

目 录

CONTENTS

第 1 章　四旋翼飞行器概述与发展现状 …… 1

1.1　空中机器人 …… 1
1.2　国内外四旋翼飞行器的研究现状 …… 3
1.2.1　国外研究现状 …… 6
1.2.2　国内研究现状 …… 11
1.3　四旋翼飞行器技术关键 …… 15
1.3.1　总体设计优化 …… 15
1.3.2　能源动力系统 …… 15
1.3.3　建立数学模型 …… 15
1.3.4　飞行控制 …… 16
1.3.5　定位、导航与通信 …… 16

第 2 章　四旋翼飞行器飞行原理 …… 17

2.1　基本原理 …… 17
2.2　四旋翼飞行器姿态表示 …… 20
2.2.1　坐标系统建立 …… 20
2.2.2　姿态解算 …… 21
2.3　平衡控制算法 …… 26
2.3.1　四旋翼飞行器的 PID 调节原理 …… 26
2.3.2　四旋翼飞行器的 PID 参数整定(利用 MATLAB 仿真确定理论参数) …… 31
2.4　滤波算法 …… 33

2.4.1 卡尔曼滤波算法 …… 33
2.4.2 互补滤波算法 …… 37

第 3 章 四旋翼飞行器常用通信协议 …… 39

3.1 常用通信接口协议 …… 39
3.1.1 SPI …… 39
3.1.2 I^2C …… 43
3.1.3 USART …… 49
3.2 常用 RC(Radio Controller)通信协议 …… 52
3.2.1 PPM …… 52
3.2.2 PWM …… 53

第 4 章 四旋翼飞行器硬件系统设计 …… 55

4.1 四旋翼飞行器系统硬件总体结构 …… 55
4.1.1 主控板 …… 56
4.1.2 外围传感器 …… 71
4.1.3 遥控器介绍 …… 93
4.2 机架的构造与电机的选用 …… 95
4.2.1 机架结构与设备安装 …… 95
4.2.2 电机与桨的选用 …… 96

第 5 章 四旋翼飞行器软件系统设计 …… 98

5.1 开发工具简介 …… 98
5.1.1 CCS …… 98
5.1.2 IAR …… 98
5.1.3 Keil …… 99
5.2 飞行控制板控制系统软件总体设计 …… 100
5.2.1 总体框架 …… 100
5.2.2 初始化 …… 100
5.2.3 中断处理 …… 101

5.2.4　飞行控制程序中的重要变量列表 …… 101
5.3　四旋翼飞行器具体功能软件实现 …… 103
5.3.1　定高飞行 …… 103
5.3.2　定点悬停与位移控制 …… 104
5.3.3　数据传输设计 …… 109
5.3.4　控制系统设计 …… 110
5.4　地面站软件设计 …… 112
5.4.1　地面站系统组成与图形化的界面操作设计 …… 112
5.4.2　与四旋翼飞行器的通信协议 …… 114
5.4.3　遥控器 PPM 信号捕获 …… 116
5.5　上位机软件设计 …… 117
5.5.1　软件功能 …… 117
5.5.2　软件实现 …… 121

第 6 章　四旋翼飞行器制作及调试方法 …… 129

6.1　飞行控制板 PCB 板制作与调试技术 …… 129
6.2　四旋翼飞行器的组装 …… 133
6.2.1　飞行器的组装步骤 …… 133
6.2.2　新组装四旋翼飞行器的第一次调试过程 …… 134
6.3　常见调试问题与解决方案 …… 137
6.3.1　PID 参数调整经验 …… 137
6.3.2　减少机械振动以提高飞行稳定性 …… 140

第 7 章　成功案例 …… 141

7.1　竞赛作品 1——2015 年(瑞萨杯)全国大学生电子设计竞赛 …… 141
7.1.1　题目要求 …… 141
7.1.2　系统方案 …… 143
7.1.3　实现方案 …… 143
7.1.4　软硬件方案 …… 145
7.1.5　测试 …… 146

7.2 竞赛作品 2——2016 年(TI 杯)上海市大学生电子设计竞赛 …… 147
7.2.1 题目要求 …… 147
7.2.2 系统方案 …… 148
7.2.3 实现方案 …… 149
7.2.4 硬件与程序方案 …… 150
7.2.5 测试 …… 152
7.3 毕业设计作品 1——目标跟踪四旋翼飞行器 …… 152
7.3.1 课题概述 …… 152
7.3.2 硬件系统设计 …… 153
7.3.3 飞行控制板软件系统设计 …… 154
7.3.4 系统板软件系统设计 …… 155
7.3.5 总结与展望 …… 157
7.4 毕业设计作品 2——基于 Kinect 的穿窗和手势控制四旋翼飞行器 …… 158
7.4.1 课题概述 …… 158
7.4.2 硬件系统概述 …… 158
7.4.3 目标物体定位和追踪 …… 161
7.4.4 路径规划及控制 …… 166
7.4.5 总结与展望 …… 175

参考文献 …… 178

第 1 章 四旋翼飞行器概述与发展现状



1.1 空中机器人

空中机器人(Aerial Robotics)最早由美国乔治亚理工大学的 Robert Michelson 提出,是指各种搭载了 GPS、机载导航设备、视觉识别设备以及无线通信设备等,能够在一定的范围内实现无人飞行的旋翼无人飞行器、无人飞艇等。空中机器人拓展了无人机的概念,在一定意义上可以将其看作是无人机的别称。空中机器人以研究微型无人飞行器为主要目标,代表了未来微型飞行器的发展趋势,是一种智能微型飞行器,代表了飞行器的最高层次。

空中机器人曾经有过多种称谓,无人飞行器(Unmanned Aerial Vehicle ,UAV)是其中的一种。自从 1917 年英国研制出世界上第一架无人飞行器,无人飞行器已经经历了无人靶机、预编程序控制无人侦察机、指令遥控无人侦察机和复合控制的多用途无人飞行器的发展过程。在越南战争、中东战争和科索沃战争中,无人飞行器卓有成效地执行了多重军事任务,包括照相侦察、散发传单、搜索信号情报、布撒雷达干扰箔条和防空火力诱饵、防空阵地位置的标识、直升机航路侦察,为武器系统提供目标定位、目标指示、目标动态监视和目标毁伤评估的实时情报。无人飞行器以突出的战绩使各国高层军事首脑对它刮目相看,对无人飞行器作为军队战斗力倍增器的作用与地位及其潜在的军事价值形成了共识,这些都为无人飞行器的迅速发展提供了强大的动力。

进入 20 世纪 90 年代,冷战结束后各国军费削减、军队裁员,迫使军方努力寻求既能完成特定任务,又花费较少的途径,这无疑对无人飞行器的发展提供了机遇。从军事侦察使用角度来看,无人侦察机是侦察卫星和有人侦察机的重要补充和增强手段,它与侦

察卫星相比，具有成本低、侦察地域控制灵活、地面目标分辨率高等特点；与有人侦察机相比，具有可昼夜持续侦察的能力，不必考虑飞行员的疲劳和伤亡问题。再加上 20 世纪末电子技术和航空航天技术的飞速发展，为无人飞行器满足军事需求在技术上提供了可能。

以上因素促使世界无人飞行器的发展进入一个新的时代，并在 20 世纪末形成三次发展的浪潮。

第一个浪潮是在海湾战争之后师级战术无人飞行器系统的发展。代表性机型有：以色列的“侦察兵”(Scout)、“先锋”(Pioneer)、“搜索者”(Searcher)；美国的“猎人”(Hunter)、“先驱者”(Outrider)，法国的“玛尔特”(Mart)、“红华”(Crecerelle)，德国的“布雷威尔”(Brevel)，加拿大的“289”(CL-289)，英国的“不死鸟”(Phoenix)，意大利的“米拉奇”(Mirach)，南非的“探索者”(Seeker)、“秃鹫”(Vulture)和俄罗斯的“熊峰”(Shmel)等。

第二个浪潮是自从美国的“捕食者”(Predator)(迪尔Ⅱ，图 1.1.1)中空长航时无人飞行器在波黑和科索沃战场使用获得成功之后，促进了中空长航时无人飞行器的研制。代表性的机型有：美国的“捕食者”(Predator)(迪尔Ⅱ)、“全球鹰”(GlobalHawk)(迪尔Ⅱ+)、“暗星”(DarkStar)(迪尔Ⅲ－)，以色列的“苍鹭”(Heron)、“赫麦斯”(Hemes)和法国的“鹰”(Eagle)、“萨若海尔”(Sarohale)等。



图 1.1.1 “捕食者”无人攻击机

目前，空中机器人的研究大都是围绕微型无人飞行器进行的，这是因为跟其他无人飞行器相比，微型无人飞行器造价低、体积较小、重量轻、飞行相对灵活、起落受限制少，能适应复杂多变的环境，更适合进行目标识别、定位等比赛项目。而且在实际应用中，由于其具有高度的灵活性和很强的适应性，微型无人飞行器在军事和民用上有着广阔的应用前景。

在军用方面，微型无人飞行器主要用于低空侦察、电子干扰等作战任务。这样不仅可减少部队在侦察过程中的伤亡，还可以大大提高作战效率。另外，微型无人飞行器还可用于目标指示及生化武器探测等。在城市作战中，微型无人飞行器的优势更为突出：能够在建筑物群中以很慢的速度飞行，以便绕过障碍物并且避免被撞倒；可以飞到大型建筑物上执行城区侦察任务；可以探测和查找建筑物内部的敌对分子或者恐怖分子，并且窃听敌方作战计划等。同时，微型飞行器也是与未来城市作战的武器发展趋势相吻

合的。

在民用方面，微型飞行器可用于通信中继、环境研究、自然灾难的监视与支援。微型无人飞行器还可用于边境巡逻、毒品禁运、农业勘测并在未来大型牧场和城区监视等方面具有广阔的市场和应用前景。

根据结构和飞行原理的不同，无人机分为两种[1]。一是固定翼无人机，分为螺旋桨式无人机和喷气式无人机，其飞行原理是在无人机高速前进的过程中，通过机翼上下气流和气压的不同，产生升力，保持无人机的飞行。二是旋翼无人机，旋翼的数量可以有一个或多个，本课题中研究的是四旋翼飞行器，旋翼无人机的原理是用一个或多个螺旋桨的高速转动产生升力，如果是多个螺旋桨，则需要通过多个螺旋桨的配合保持平衡，并控制无人机 6 个自由度方向的运动。旋翼无人机最大的特点就是可以垂直起降和悬停，与固定翼无人机相比，有很好的机动性，对场地要求也较小，可扩展功能十分广泛。

四旋翼飞行器与其他多旋翼飞行器相比较，其优点在于，机架易于制作，不容易损坏且容易维修，单位体积拥有更大的动力等。另外，四旋翼无人机有能力在狭窄的空间内实现各种功能。所以，因为其广阔的发展前景，近年来，不仅吸引了科研人员关注，商品市场上也出现了越来越多的无人机。目前，市场上已经有很多民用的无人机产品，主要用于航拍和娱乐方面，户外手控飞行技术已经十分成熟。

1.2　国内外四旋翼飞行器的研究现状

四旋翼飞行器在诞生之初，由于传感器、微处理器等的发展还不是特别成熟，其发展受到了许多局限，也迟迟不能进入实用阶段。世界上第一架四旋翼飞行器 Gyroplane No. 1（如图 1. 2. 1 所示）在 1907 年成功升空，它是由法国 Breguet 兄弟制造的多旋翼飞行器，但飞行稳定性非常差[2]。

1922 年，George de Bothezat 制作了“飞天章鱼”（如图 1. 2. 2 所示），设计者 Bothezat 直接着手做原始尺寸的飞行器，该飞行器身长、宽都为 65 英尺（ft，1ft≈0. 3048m），高为 10 英尺，机重 3600 磅（lb，1lb≈0. 454kg），使用了 180HP（1HP≈745W）的发动机，虽然只能成功飞行 6 英尺，并且只能飞行 1min，但这无疑是一个巨大的进步。

旋翼飞行器在 20 世纪 50 年代被应用于陆军机构，开始试验各种垂直起飞降落的方案，Curtiss-Wright VZ-7（如图 1. 2. 3 所示）使用了杠杆燃气涡轮机来提高性能，但同时也



图 1.2.1　Gyroplane No. 1 实物



图 1.2.2　“飞天章鱼”实物

增加了飞机重量，1950—1960 年期间实现了相对稳定的飞行，但由于它没有达到要求的高度和速度，所以测试试验没有进一步施行。



图 1.2.3　Curtiss-Wright VZ-7

在美国 20 世纪 90 年代早期，工程师 Mike Dammar 自己开发了电池给飞行器供电，在其加入 Spectrolutions 公司工作之后将该飞行器命名为 Draganflyer（如图 1.2.4 所示），后来加拿大分销商进行了销售。2006 年又安装了稳定的航拍视频系统，称为“天眼”。

几十年来，随着微控制器技术的巨大突破，以及新型材料、微机电系统（MEMS）、



图 1.2.4　Draganflyer

微惯导(MIMU)以及飞行控制等技术的进步,四旋翼飞行器又逐渐走入人们的视野。四旋翼飞行器在布局上属于非共轴式的蝶形,四只旋翼产生相互抵消的反扭力矩,具有结构简单、稳定性高、易于控制、成本相对较低,能实现垂直起降(VTOL)等特点。与之类似,还有如六旋翼、八旋翼飞行器等。

VC200 是德国 E-vovo 公司研发的一款多旋翼飞行器(如图 1.2.5 所示),主题机身采用碳纤维复合型材料,拥有 18 个电机、旋翼,采用 6 组电池(每组为 3 台发动机供电),能维持约 20min 的飞行。



图 1.2.5　VC200 多旋翼飞行器

四旋翼飞行器在军事领域、自然和民用等领域具有广阔的应用前景,如灾害搜救、军事打击、地形勘探、广告航拍、航模玩具等。目前,亚马逊、Google、DHL 等公司都在研发无人机快递业务服务,其中亚马逊无人送货机如图 1.2.6 所示。其飞行器也是运用了多旋翼飞行器,相信不久的将来,将不用快递员进行送货,这将大大减少人力投资,并在很

大程度上改善人们的生活。



图 1.2.6 亚马逊无人送货机

四旋翼飞行器的应用还进入了餐饮行业，餐厅用飞行器为用户送上热腾腾的餐食。英国达美乐分公司通过跟创意公司合作开发了一架名为“达美乐直升机”的送餐飞行器。它能在 10min 之内将食物送到 4 英里（1 英里≈1.61km）之外，极大地提高了送餐效率。同样地，一家名为“Yo! Sushi 寿司”的餐馆也提出了用飞行器进行送餐的全新方式——服务员通过控制 iPad 来操控四旋翼飞行器，将承载食物的餐盘送到顾客的餐桌上（如图 1.2.7 所示）。



图 1.2.7 利用四旋翼飞行器进行送餐

因此，由于四旋翼飞行器所具备的以上特点及其广阔的应用前景，人们对研究四旋翼飞行器的热情被重新点燃了。

1.2.1 国外研究现状

国外许多大学和商业公司在近十几年间对四旋翼飞行器进行了研究和探索。下面

将对他们的工作加以介绍。

自2002年开始，宾夕法尼亚大学(UPenn)提出了基于视觉反馈的直升机控制系统。研究人员通过使用HMX-4这一商业模型开发了自己的四旋翼飞行器(如图1.2.8所示)。其飞行器控制系统是通过系统中微控制器和远程遥控来实现的，远程PC首先对传感器传输的数据进行分析处理，然后回送控制信息。其四轴控制系统中携带3种传感器模块，用于实现闭环的增稳控制。同时整套系统中，地面安放了用于主监控的摄像机，并在飞行器底部设计了彩色符号来为摄像机定位，利用这些符号计算出飞行器的相对位移，地面PC即可算出四旋翼飞行器的空间旋转角度和水平位移。这一团队的研究成果也多次出现在TED演讲、优酷视频等中，自主悬停时使用基于模型的线性反馈控制，而在穿越障碍、自主飞行时与视觉反馈控制相结合，其研究重点已经向多机协作和自主飞行倾斜。



图1.2.8　宾夕法尼亚大学Kumar Lab设计的四旋翼飞行器

Mesicopter(如图1.2.9所示)是斯坦福大学(Stanford)的研究小组在NASA支持下，为研究微型旋翼飞行器技术而设计的试验装置[3]。机身为16mm×16mm的方形框架；旋翼直径为1.5cm，厚度为0.08mm；电机直径为3mm，质量为325mg。Mesicopter飞行器研究组设计了应用于自主飞行器控制研究(STARMAC)的测试平台，其控制系统包含：惯性测量单元(IMU)、微控制器、超声波测距传感器、GPS定位导航单元以及蓝牙通信模块，主要实现了姿态估测和数据通信；在控制端则设计了PC和遥控器操作装置。惯性测量单元从传感器中获得姿态数据后估测出当前飞行器的高度及其运动速度，然后输出这两组数据，由于飞行器在升力变化时会产生较大震动，因此两组数据混有较大的噪声，导致估测准确性降低；同时超声波传感器的数据也不可靠，所以需要卡尔曼滤波器

首先去噪,再进行高度估测。系统中还加入了红外传感器来协助完成飞行器的距离测量任务。系统中还设计了另一个卡尔曼滤波器用来将计算出的 GPS 数据、速度数据以及高度信息融合在一起,计算出飞行器的位置和速度信息。

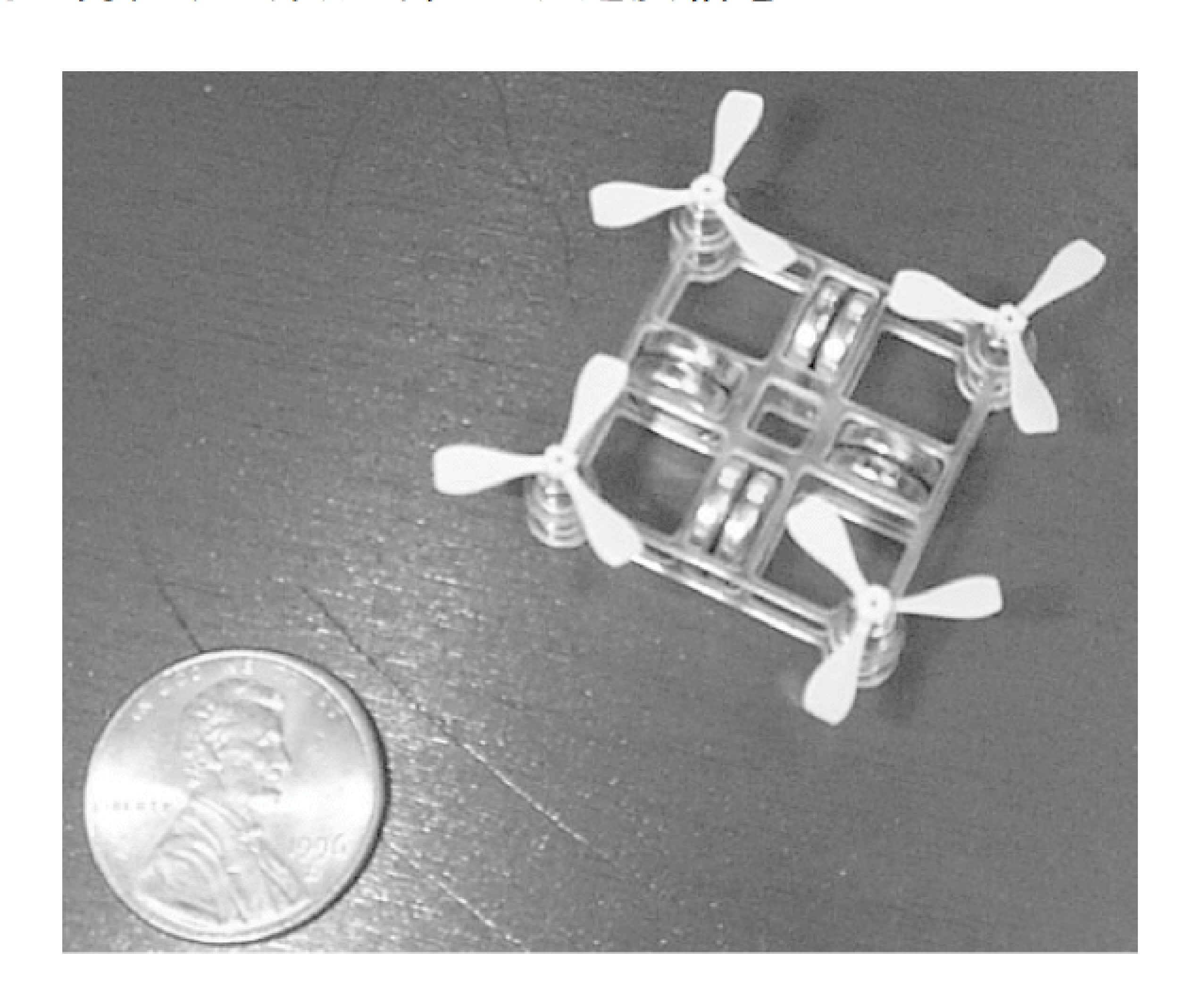


图 1.2.9 斯坦福大学的 Mesicopter

瑞士洛桑联邦理工学院于 2003 年开始研发了 OS4 微型四旋翼飞行器(如图 1.2.10 所示),试验了各种不同的控制算法,对于最优控制理论、飞行器自主飞行和避障等做了比较深入的研究。OS4 是 EPFL 自动化系统实验室开发的一种电动小型四旋翼飞行器,其研究的重点是机构设计方法和自主飞行控制算法,目标是要实现室内和室外环境中的完全自主飞行。

目前,该项目已经进行了两个阶段。OS4 Ⅰ最大长度约 73cm,质量为 235g;它使用了 Draganflyer 的旋翼和十字框架,4 个 Faulhaber 1724 电机,以及一个 Xsense 的 MT9-B 微惯性测量单元。至 2004 年,OS4 Ⅰ已经分别基于多种控制算法(如 PID、LQ、Backstepping、Sliding-mode),实现了飞行器姿态控制。OS4 Ⅱ的机身最大长度为 72cm,重为 520g;机载 230g 的锂电池,提供自主飞行 30min 的能量。它与 OS4 Ⅰ的区别主要有:使用了桨叶面积更大的新旋翼;使用了更轻、功率更大的 LRK 无刷电机 BLDC;使用皮带减速装置代替了电机减速箱;控制器、传感器、电池和电机驱动模块等都直接安装在机体上,不再由机体外部提供。

此外麻省理工学院的 Robust Robtics 和佐治亚理工大学的 GTMARS 等研究组也都对四旋翼飞行器的建模、飞行、视觉控制、周围环境监测等做了很多研究工作。



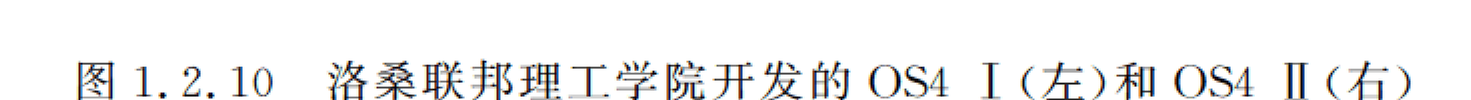


图 1.2.10　洛桑联邦理工学院开发的 OS4 Ⅰ(左)和 OS4 Ⅱ(右)

在商业领域，法国巴黎的 Parrot 公司在 2010 年的国际消费类电子产品展览会(Consumer Electronics Show)上展示了一款名为 AR. Drone 的双摄像头航模玩具(如图 1.2.11 所示)，以其创新的玩法获得了 2010 年国际消费电子展电子游戏硬件创新奖。AR. Drone 类似于一款无人驾驶直升机，拥有 4 个独立螺旋桨，用户可以使用 iPhone、iPod Touch 对其进行飞行控制操作，能够实现自动悬浮定位、自动导航驾驶、手机无线遥控、高清实时航拍等有趣而复杂的功能。



图 1.2.11　AR. Drone 四轴玩具飞机

拉脱维亚直升宇航工业公司(Helico Aerospace Industies)所设计的 AirDog 飞行器(如图 1.2.12 所示)是一款专为极限运动所打造的跟拍器。AirDog 是一款专有航拍器，支持索尼以及 GoPro 摄像机，折叠起来小到可以放进背包，其最独特的地方是可实现全自动。AirDog 还配置了一个名为 AirLeash 的腕带控制器，用户只须将其绑在手臂上就能操作控制 AirDog 的起飞和降落，在行动过程中，AirDog 还利用了回转仪稳定技术确保被拍摄物体始终能在拍摄画面里。该设备可以预先设定飞行模式，这样就能确保在不同的体育运动中拍摄到最佳效果的影片，这些体育运动包括极限脚踏越野车(BMX)、极限滑雪、冲浪、越野摩托车、水上滑板等，可以通过手机 APP 进行设定。

四旋翼飞行器在其他领域的潜力也不断被挖掘出来。苏黎世联邦理工学院(ETH)的法比奥格·拉马里奥与马提亚斯·科赫勒教授运用数控加工技术控制的四旋翼飞行器操控传统的砖块堆砌起一座高 6m、直径 3.5m 的 1∶100 的名为“垂直村落”的建筑模型(如图 1.2.13 所示),模型本身的建造方法也正是类比着设计中垂直向上“生长”的高层“村落”。整个过程完全由多架四旋翼飞行器独立完成,使用了超过 1500 块聚苯乙烯泡沫制作的砖块,几架四轴直升机像蜜蜂一样回旋飞舞,抓起一块块砖头,精准定位后将砖头一块块放下,层层叠叠地砌出一个曲线形状的大尺度构筑物。



图 1.2.12 AirDog



图 1.2.13 四旋翼飞行器堆砌出来的建筑模型

四旋翼飞行器的主要优点如下。

① 拥有简单的机械结构,没有传统直升机复杂的旋翼控制机构。

② 拥有稳定的飞行姿态。飞行器控制系统通过传感器采集飞行姿态数据,实时监测和控制飞行姿态,可以实现飞行器保持平稳的飞行。与传统单旋翼直升飞机相比,四旋翼飞行器的飞行更为稳定。

③ 容易将体积做得很小,可以实现自主飞行器的小型化和微型化。微型四轴器调试相对容易简单,而制作完成之后,执行任务的地点也更加多样化。

④ 飞行器拥有 4 个电机,具有更大的载重力。

此外，四旋翼飞行器还具有成本低、噪声小的优点，比单翼直升机或者其他垂直起降机其更易于维护。因此，旋翼无人机广泛应用于民用和军事领域，如救援、侦察、勘探等方面。

1.2.2　国内研究现状

国内对于四旋翼飞行器的研究起步较晚，但是发展迅速，最早的研究开始于 2000 年左右。

2001 年，上海大学通过利用已有的相关技术，初步研究和设计四旋翼飞行器飞行控制，并实现飞行试验。重点对整个飞行器做了样机设计，包括控制算法设计、控制系统软件设计和硬件设计。为实现自主控制奠定了基础。

国防科技大学在 2004 年开始研究微型四旋翼飞行器，做了一系列的建模和实践，并在 2006 年完成了对四旋翼飞行器的原型样机的设计，根据 Quad-Rotor 飞行原理和牛顿—欧拉方程建立了四旋翼飞行器的动力学模型，设计了基于模糊滑模控制理论的姿态控制算法，并且通过试验仿真验证了四旋翼飞行器姿态控制算法的有效性，实现了基于 Quad-Rotor 的姿态增稳控制。

随后，哈尔滨工业大学建立了四旋翼飞行器的动力学模型和运动学模型，并针对四旋翼飞行器系统的线性模型的不确定性，基于回路成形法设计实现了鲁棒控制器，并基于四旋翼飞行器的非线性模型进行了仿真分析和研究，取得了比较好的控制效果。

此外，南京航空航天大学、西北工业大学等高校也相继进行了较多的试验和探索，广东的模型厂商也推出了多款小型四旋翼飞行器。但四旋翼飞行器要走向成熟和实用，还面临着诸多关键技术的挑战。

除了各大高校的研究攻关，多旋翼飞行器各方面技术的进步和成熟也离不开诸多消费级无人机公司的推动，而以大疆创新科技有限公司为首的各个创业公司则成为了行业的引领者。

深圳市大疆创新科技有限公司(DJI)成立于 2006 年，是全球领先的无人飞行器控制系统及无人机解决方案的研发和生产商，客户遍布全球 100 多个国家。大疆公司致力于通过持续的创新为无人机工业、行业用户以及专业航拍应用提供性能最强、体验最佳的革命性智能飞行控制产品和解决方案。

2013 年 1 月，大疆公司发布“精灵”(如图 1.2.14 所示)，这一产品一举撬动了非专业无人机市场，使得大疆公司在市场上站稳脚跟，为大疆公司成为这一行业引领者打下了

坚实的基础。现在大疆公司已经形成了完整的产品线，除“精灵”系列外，大疆还将2014年11月面世的InSPIre(悟)(如图1.2.15所示)系列定位为“一体化空中影像系统”，此系列的目标群体是高端电影、视频创作者；筋斗云系列飞行平台是为资深航拍爱好者开发的一系列专业级载机，具有便携易用、安全稳定等特点，主要应用于专业航拍领域；MG-1S农业植保机集成大疆最新技术，可实现作业规划、飞行实时管理和工作统计，不仅是一款高性能无人机，还为消费者提供了一整套智能便捷的农业植保解决方案；Mavic主打小体积、远距离图传和长时间续航，巧妙的折叠设计让消费者能轻装上阵，尽情享受飞行与拍摄的乐趣。





图1.2.14　大疆“精灵”系列无人机



图1.2.15　大疆InSPIre(悟)系列无人机

PowerEgg无人机(如图1.2.16和图1.2.17所示)是臻迪推出的首款消费级无人机，此项目于2014年6月立项，2016年8月进行京东众筹，受到广泛关注，并在短时间内众筹成功。通过这款融合了科技与艺术的飞行机器人，臻迪希望能让更多的人体验到航拍飞行的乐趣，所以PowerEgg通过技术创新和设计创新，主打易用性、便携性及安全性等诸多功能，希望帮助用户解决痛点。这款无人机最大的特点就是外形被设计成了蛋形，机臂可以完全折叠，一体化的造型十分方便用户携带；而从操控上，臻迪希望除了专业的航拍手、摄影爱好者和一些航模爱好者之外，能有更多对无人机感兴趣的小白用户也可以享受到飞行的乐趣。因此，PowerEgg除了标准版的遥控器之外，还特别推出了体感遥控器，通过体感遥控器可以实现一键悬停、一键拍照、一键跟踪环绕等功能，简单易上手，甚至连小孩也能操作。

2016年4月27日，ZeroRobotics公司正式发布Hover Camera(如图1.2.18和图1.2.19所示)，这是全球首部真正意义上的安全、易用、便携式跟拍无人机，这款无人机的质量只有238g，却拥有1300万像素摄像头，可以拍摄4K视频，同时还可以提供720像素高清实时数据流，这些数据流可以通过WiFi传到智能手机。这款无人机的特别之





图 1.2.16　PowerEgg 无人机折叠图



图 1.2.17　PowerEgg 无人机展开图

处在于其可以悬停在空中完成这些任务，用户可以通过智能手机指挥其行动。Hover Camera 也支持悬停旋转全景拍摄。它采用碳纤维外壳以及创新性的可折叠式设计，大大增强了产品的安全性和便携性。无须费力学习或组装，只要展开机翼并开机，Hover Camera 便会从用户的指尖稳稳起飞，以独特视角拍摄视频或照片。这款产品主打便携，折叠后的收纳尺寸为 18cm×13cm，同时具备体感操控、自动跟随、指尖放飞和空中摘取等功能。



图 1.2.18　Hover Camera 无人机折叠图



图 1.2.19　Hover Camera 无人机展开图

除了这些明星产品，近 10 年来，国内同样涌现出了一批拥有自己核心技术的无人机公司，如亿航、零度、极飞和易瓦特等。

2007 年成立的极飞是我国最早开始研发多旋翼无人机的公司之一，如今也是着重农业领域的无人机公司之一。该公司研制的“农飞 P20 植保无人机系统(XPlanet)”(如图 1.2.20 所示)帮助新疆棉农喷洒农药，将无人机真正应用到实处。同时，极飞无人机还与物流公司友好合作，举办了轰动一时的淘宝无人机送货服务。

虽然零度智控公司成立于 2007 年，但是 2011 年才开始研制多旋翼无人机，2015 年



图 1.2.20　极飞植保无人机

设计消费级无人机。第一款消费级无人机产品——Xplorer 标准版(如图 1.2.21 所示)售价仅 2499 元,让拥有炫酷的四旋翼飞行器不再是梦想,因此也被称为“价格屠夫”。该公司无人机主要应用于航拍领域,在《智取威虎山》和《天河》等电影中均有应用,如今已经占据 1/3 的航拍市场。

亿航科技于 2014 年在广州成立,可以说是无人机行业的新生力量,其实力不容小觑。旗下第一款无人机 Ghost(如图 1.2.22 所示)仅需要一款手机 APP 就可轻松实现起飞、返航、下降等一系列操作。该无人机以 80 万美元的成绩创下了中国海外众筹项目的新纪录。如今有报道称一家名为 Lung Biotechnology 的生物科学公司计划与亿航科技合作,利用无人机运送移植器官,虽然有些不可思议,但是也肯定了亿航无人机的发展与努力。



图 1.2.21　零度 Xplorer 无人机



图 1.2.22　亿航 Ghost 无人机

1.3　四旋翼飞行器技术关键

1.3.1　总体设计优化

进行微小型四旋翼飞行器总体设计时，需要遵循以下原则：质量轻、尺寸小、速度快、能耗和成本低。但这几项原则相互之间存在着制约与矛盾，因此进行四旋翼飞行器总体设计时，首先要根据性能和价格选择合适的机架材料，尽可能地减轻飞行器质量；其次要综合考虑质量、尺寸、飞行速度和能耗等因素，确保实现总体设计的最优化。

1.3.2　能源动力系统

动力装置包括旋翼、微型直流电机、电调等，能量由机载电池提供。四旋翼飞行器的质量是影响其尺寸的主要因素，而动力与能源装置的质量在整个机体质量中占了很大比例。对于 OS4 Ⅱ，该比例就高达 75%。因此，研制更轻、更高效的动力与能源装置是进一步微小型化四旋翼飞行器的关键。另外，动力装置产生升力时消耗了绝大部分机载能量。例如，OS4 Ⅱ 的电能有 91%被动力装置消耗。要提高飞行器的效率，关键在于提高动力装置的效率。除尽量提高机械传动效率外，还必须选择合适的电机与减速比，在兼顾最大效率和最大输出功率两项指标的前提下，将电机工作点配置在推荐运行区域内。

1.3.3　建立数学模型

为实现对微小型四旋翼飞行器的有效控制，必须准确建立其在各种飞行状态下的数学模型。但在飞行过程中，它不仅同时受到多种物理效应的作用(空气动力、重力、陀螺效应和旋翼惯量矩等)，还很容易受到气流等外部环境的干扰。因此，很难建立有效、可靠的动力学模型。此外，所使用的旋翼尺寸小、质量轻、易变形，很难获得准确的气动性能参数，也将直接影响模型的准确性。

建立四旋翼 MAV 数学模型时，还必须深入研究和解决低雷诺数条件下旋翼空气动力学问题。微型飞行器空气动力学特性与常规飞行器有很大的不同，当前许多空气动力学理论和分析工具均不适用，需要发展新的理论和研究手段。

1.3.4 飞行控制

四旋翼飞行器是一个具有六自由度(位置与姿态)和 4 个控制输入(旋翼转速)的欠驱动系统(Underactuated System),具有多变量、非线性、强耦合和干扰敏感的特性,使得四旋翼飞行器的设计变得非常困难。此外,控制器性能还将受到模型准确性和传感器精度的影响。姿态控制是整个飞行控制的关键,因为四旋翼飞行器的姿态与位置存在直接耦合关系(俯仰、横滚直接引起机体向前后、左右移动),如果能精确控制飞行器姿态,则采用 PID 控制律就足以实现其位置与速度控制。国际相关研究都着重进行了姿态控制器的设计与验证,结果表明,尽管采用非线性控制律能够获得很好的仿真效果,但由于对模型准确性有很强的依赖,其实际控制效果反而不如 PID。因此,研制既能精确控制飞行器姿态,又具有较强抗干扰和环境自适应能力的姿态控制器是四旋翼飞行器研究的关键。

1.3.5 定位、导航与通信

微小型四旋翼飞行器主要面向近地面环境,如城区、森林、隧道和室内等。但是,目前还存在定位、导航与通信方面的问题。一方面,在近地面环境中,GPS 常常不能正常工作,需要综合惯导、光学、声学、雷达和地形匹配等技术,开发可靠、精确的定位与导航技术;另一方面,近地面环境地形复杂,干扰源多,当前通信链技术的可靠性、安全性和抗干扰性还不能满足实际应用的需求。因此,研制体积小、质量轻、功耗低、稳定可靠和抗干扰的通信链对微小型四旋翼飞行器技术(尤其是多飞行器协同控制技术)的发展而言是十分关键的。

第 2 章



四旋翼飞行器飞行原理

2.1 基本原理

四旋翼飞行器通过改变自身 4 个旋翼的转速，可以比较灵活地进行各种飞行动作。其主要依据的运动原理是力的合成与分解，以及空气转动扭矩的反向性。四旋翼飞行器通常有两种模式，即“十”字形模式与 X 形模式。

需要注意的是，相邻的两个旋翼的转动方向相反，而在对角线上的两个电机的转动方向相同。例如，图 2.1.1 中的 X 形模式，1、4 电机是顺时针方向转动，而 2、3 电机是逆时针方向转动。这样一来，为了保证它们产生的升力都是向上的，1、4 电机需要使用的螺旋桨是“正桨”，一般指顺时针方向转动能产生向上升力的桨；而 2、3 电机使用的螺旋桨是“反桨”，即逆时针方向产生向上升力的桨。在装配四轴时，也应注意 4 个旋翼都是“向下吹风”的，以便均提供向上的拉力。

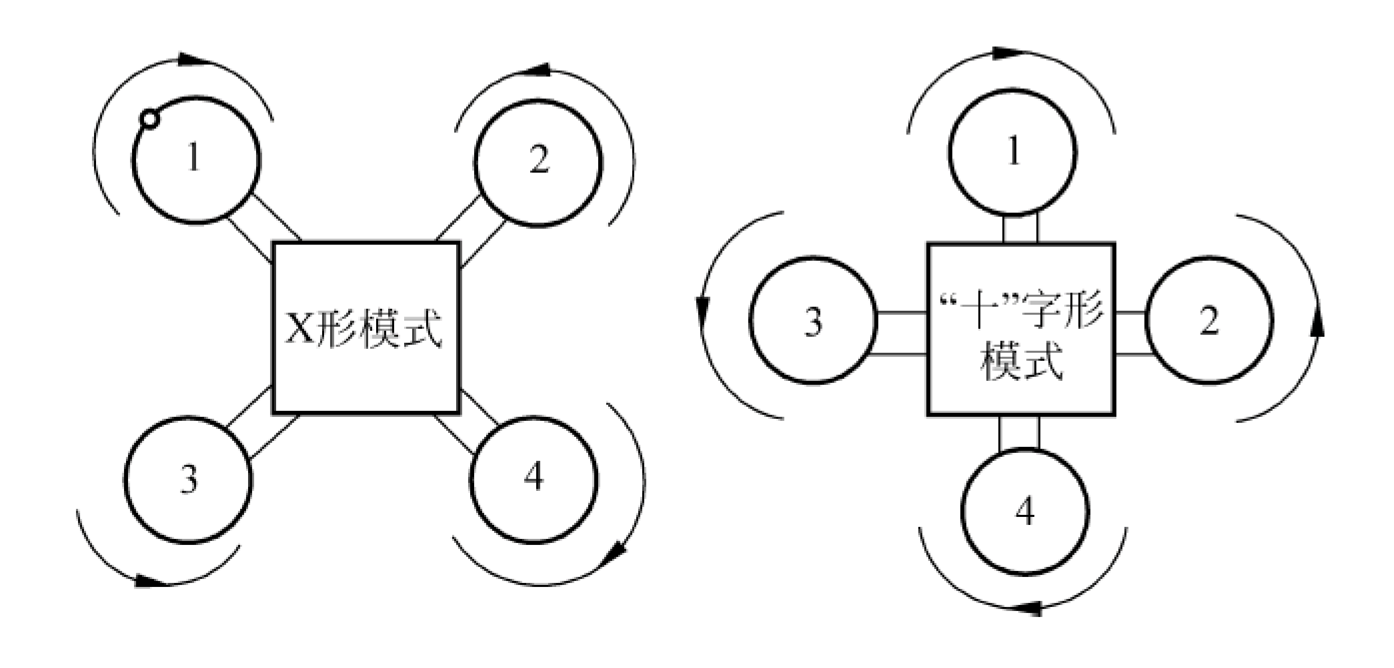


图 2.1.1　X 形模式与“十”字形模式

这样做的原因是旋翼在旋转时会产生反扭矩。例如，顺时针方向转动的桨在转动时，空气会产生使得四轴逆时针方向转动的反向扭矩。而当1、4同方向，2、3同方向的时候，这两个扭矩就恰好抵消掉，使得四轴在偏航方向能保持平衡，不至于出现自旋转。

四旋翼飞行器可以分别沿着机体的 X、Y、Z 三个轴进行旋转或者平移运动，因此在每个轴向上有两个自由度。四旋翼飞行器有6种基本飞行动作。

1. 升降运动

升降运动(如图2.1.2所示)实际上就是飞行器在 Z 轴方向的上下运动。假设四旋翼飞行器处于平稳状态，4个旋翼转速完全一样。此时同时使4个旋翼转速增加，可以让升力克服机体重量，使得机体垂直向上运动；反之，同时减小4个旋翼的转速，可使机体垂直下降。当4个旋翼处于某一转速时，升力和机体重力相等，此时飞行器处于平稳悬停的状态。

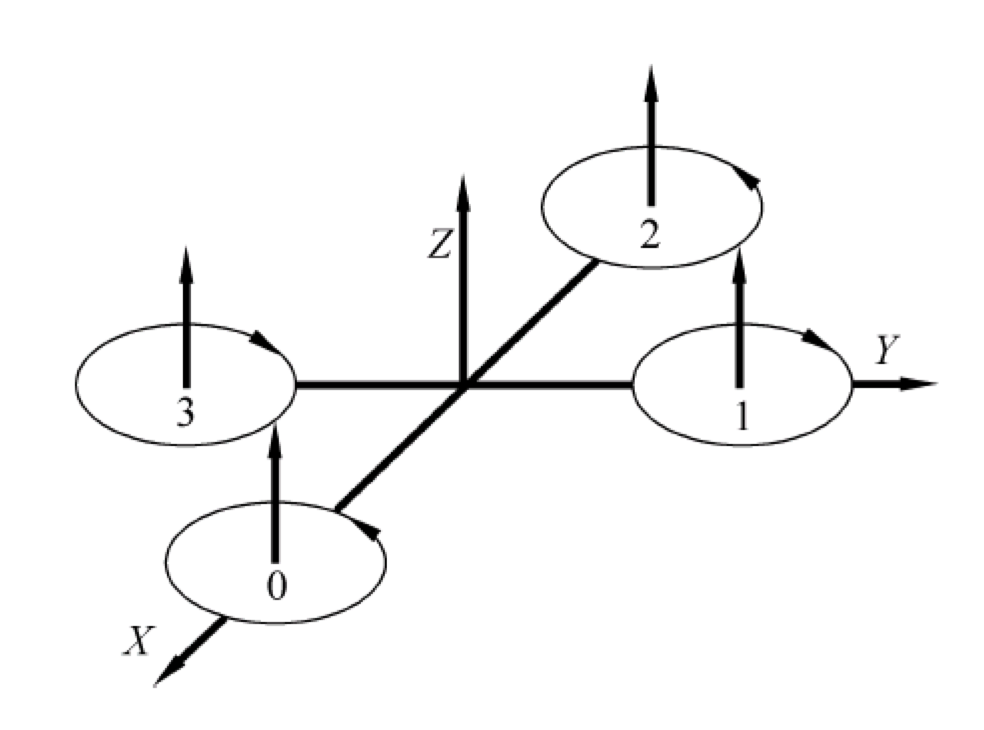


图2.1.2　升降运动

2. 俯仰运动

假设以 X 轴正方向为飞行器前进方向(电机0为前方)，俯仰运动是飞行器以机体坐标 Y 轴为中心轴的一个转动。如图2.1.3所示，以仰动作为例，在保持电机1、3转速不变的同时，电机0的转速增加使升力加大，电机2的转速减小使升力降低。飞行器会以机体 Y 轴(电机1、3所在轴)为转轴产生一个转动，电机0上升，而电机2下降，这就是飞行器的仰动作；反之，保持电机1、电机3的升力不变，让电机2转速增加，电机0转速减小，将实现飞行器俯的动作。俯仰动作需要通过控制飞行器前后方向上的两个电机的速率来实现。俯仰动作实质上还会影响飞行器的前进和后退动作。

3. 横滚运动

横滚运动与俯仰动作原理相同，如图2.1.4所示，保持电机0和电机2的转速不变，

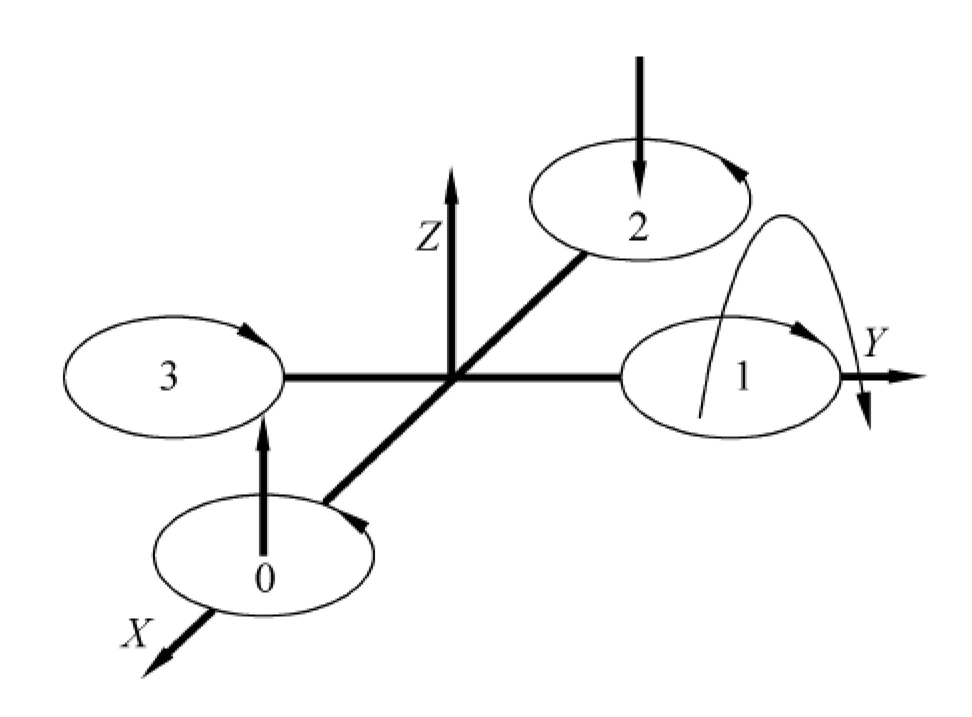


图 2.1.3　俯仰运动

分别改变电机 1、3 的转速，可使机身以机体坐标的 X 轴为旋转轴旋转。横滚动作实质上还会影响飞行器的左移和右移动作。

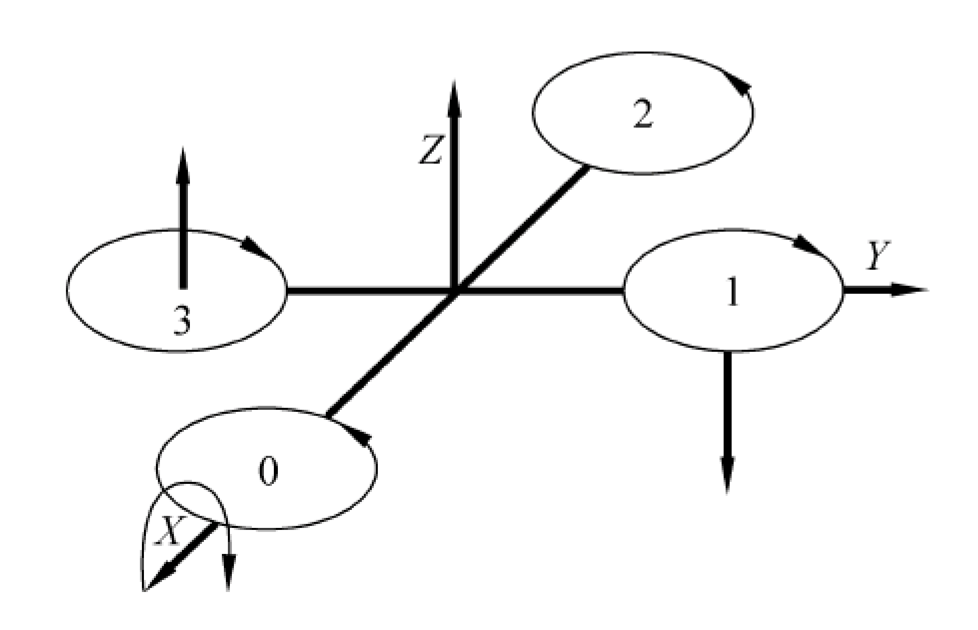


图 2.1.4　横滚运动

4. 偏航运动(自旋)

偏航(转向)运动实际上是机体绕自身坐标 Z 轴的一个自旋转的过程。从单个旋翼来看，旋转中的旋翼会对机体有一个反扭矩。如果飞行器为单旋翼，在旋翼转动时，机身会朝反方向旋转。因此，多轴飞行器的旋翼均为偶数个，而且正反转各一半。这样在飞行中，正反转旋翼的反扭矩会相互抵消。四旋翼飞行器的偏航运动实质上是通过 4 个旋翼转速不同而使得扭矩不平衡的结果，就是使得机身绕机体坐标 Z 轴旋转。如图 2.1.5 所示，当电机 0 和电机 2 转速上升，同时电机 1 和电机 3 转速下降，电机 0、2 的反扭矩大于电机 1、3 的反扭矩，机身会以机体 Z 轴为转轴以电机 0、2 旋转方向的反方向旋转；反之，当电机 1、3 的转速上升，电机 0、2 的转速下降时，机身会朝与电机 1、3 转动相反的方向旋转。

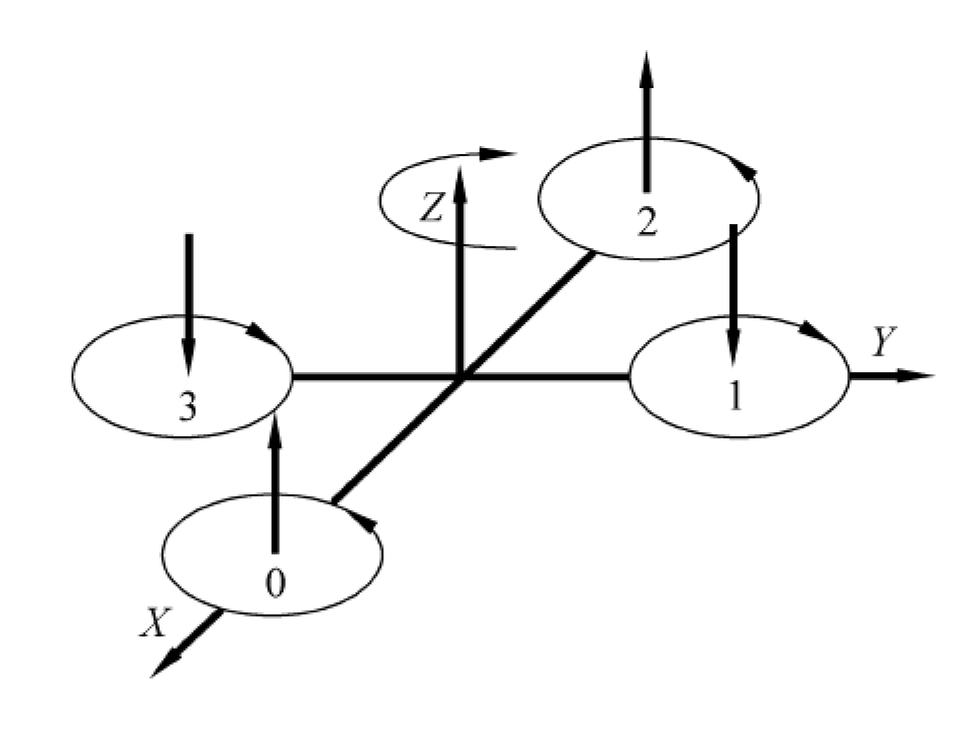


图 2.1.5 偏航运动

2.2 四旋翼飞行器姿态表示

2.2.1 坐标系统建立

为了获得四旋翼飞行器控制系统的数学模型，首先建立两个基本坐标系，即惯性坐标系 $E(O_{XYZ})$和飞行器坐标系 $B(O_{XYZ})$，如图 2.2.1 所示。通常在大地上要建立一个全局坐标系(Global Coordinate System，GCS)，用来对整个多体系统提供一个统一的参照坐标系。在物体上要建立一个局部坐标系，称为 BCS(Body Coordinate System)，一方面用来描述物体在 GCS 内的位置和姿态，另一方面为物体上的点或其他坐标系提供局部的确定位置和姿态的标准。

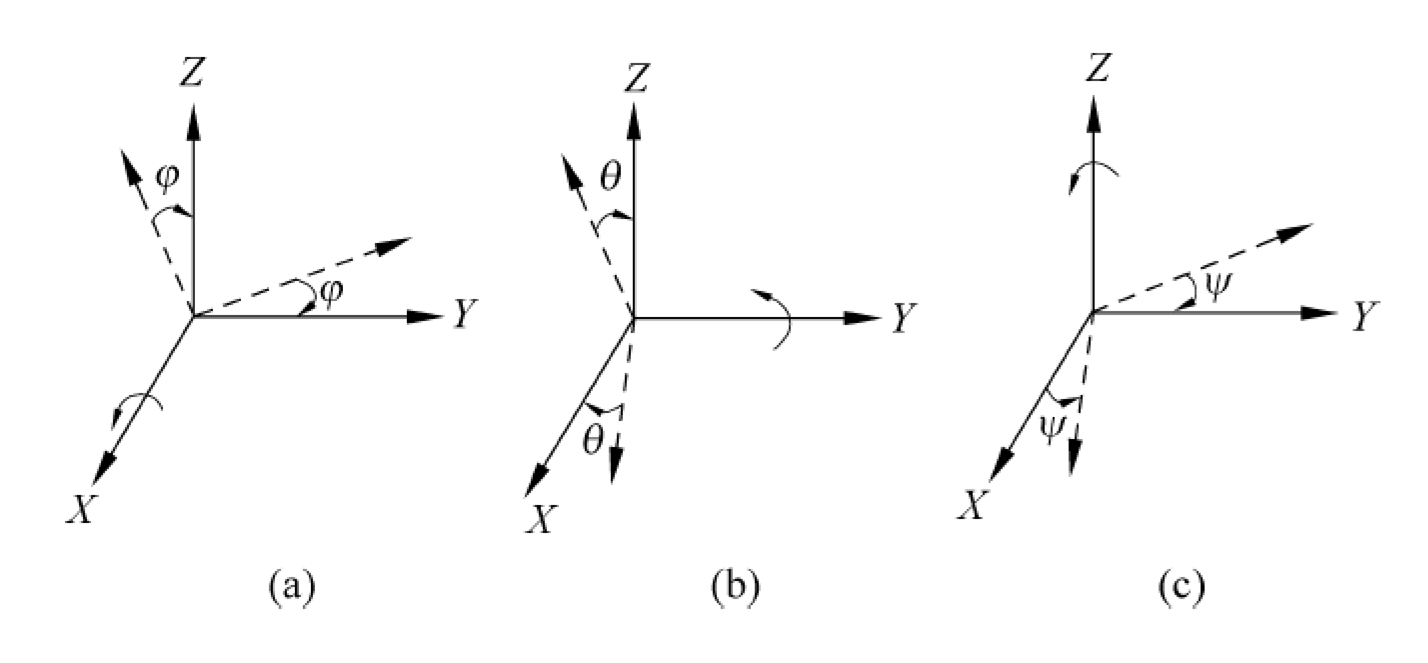


图 2.2.1 基本的坐标系

分别定义欧拉角如下。

① 航角：O_X 在 O_{XY}平面的投影与 X 轴夹角 φ。

② 仰角：O_Z 在 O_{XZ}平面的投影与 Z 轴夹角 θ。

③ 滚角：O_Y 在 O_{YZ} 平面的投影与 Y 轴夹角 φ。飞行器坐标系到惯性坐标系的转换矩阵如下。

$$\boldsymbol{R}=R_xR_yR_z=\begin{bmatrix}\cos\psi\cos\phi & \cos\psi\sin\phi & \cos\psi\sin\theta\cos\phi+\sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi & \sin\psi\sin\theta\cos\phi-\sin\phi\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi\end{bmatrix} \quad (2.2.1)$$

在建立上述坐标系的基础上，且无风低速飞行的情况下，忽略阻力系数，可以得到以下数学模型，即

$$\ddot{x}=(\cos\psi\sin\theta\cos\phi+\sin\psi\sin\phi)U_1 \mid m$$

$$\ddot{y}=(\sin\psi\sin\phi\cos\phi\text{-}\cos\psi\sin\phi)U_1 \mid m$$

$$\ddot{z}=(\cos\phi\cos\theta)U_1 \mid m$$

$$\ddot{\phi}=[lU_2+\dot{\theta}\dot{\psi}(I_y-I_z)] \mid I_x$$

$$\ddot{\theta}=[lU_3+\dot{\phi}\dot{\psi}(I_z-I_x)] \mid I_y$$

$$\ddot{\psi}=[lU_4+\dot{\phi}\dot{\theta}(I_x-I_y)] \mid I_z$$

其中，各字符代表含义如表 2.2.1 所示。

表 2.2.1　字母定义

变量	含　　义
l	旋翼中心到坐标原点的距离
m	飞行器质量
U_1	垂直速度控制量
U_2	翻滚输入控制量
U_3	俯仰控制量
U_4	偏航控制量

四旋翼飞行器具有 6 个自由度，通过 IMU 惯性测量单元实时测量飞行器 3 个轴向的旋转角速度和加速度，进行姿态解算（即对数据进行处理融合后得到当前飞机的姿态信息），控制飞行器的 4 个输入控制量，飞行器即可达到平衡飞行的目的。

2.2.2　姿态解算

姿态解算是捷联式惯性导航系统的关键技术，通过姿态矩阵可以得到载体的姿态和导航参数计算需要的数据，是捷联式惯导算法中的重要工作。载体的姿态和航向体现了

载体坐标系与导航坐标系之间的方位关系,确定两个坐标系之间的方位关系需要借助矩阵法和力学中的刚体定点运动的位移定理。通过矩阵法推导方向余弦表,而刚体定点运动的位移定理表明,定点运动刚体的任何有限位移都可以绕过定点的某一轴经过一次转动来实现。目前,描述动坐标相对参考坐标系方位关系的方法有多种,可简单地将其分为 3 类,即三参数法、四参数法和九参数法。三参数法也称欧拉角法,四参数法通常指四元数法,九参数法称为方向余弦法。

一个动坐标系相对参考坐标系的方位可以完全由动坐标系依次绕 3 个不同的轴转动的 3 个角度来确定。例如,载体坐标系作为动坐标系,导航坐标系作为参考坐标系,则姿态角即为一组欧拉角,按一定的转动顺序即可得到导航坐标系到载体坐标系的关系。

$$\begin{bmatrix}\dot{\gamma}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\frac{1}{\cos\theta}\begin{bmatrix}\cos\theta & \sin\gamma\sin\theta & \cos\gamma\sin\theta\\ 0 & \cos\theta\cos\gamma & -\sin\gamma\cos\theta\\ 0 & \sin\gamma & \cos\gamma\end{bmatrix}\begin{bmatrix}\omega E_{bb}^{b}X\\ \omega E_{bb}^{b}Y\\ \omega E_{bb}^{b}Z\end{bmatrix} \tag{2.2.2}$$

式中,γ、θ、ψ 分别对应横滚(Roll)、俯仰(Pitch)、航向(Yaw)3 个姿态角,根据欧拉角微分方程,由角速度可以求解 3 个姿态角。欧拉角微分方程式只有 3 个,但每个方程都含有三角函数的运算,计算速度慢,且方程会出现“奇点”,使方程式退化,故不能全姿态工作[4]。因此,欧拉角法难以广泛用于工程实践,且实时计算困难。方向余弦法避免了欧拉法的“奇点”现象,但方程的计算量大,工作效率低。随着飞行运载体导航控制系统的迅速发展和数字计算机在运动控制中的应用,控制系统要求导航计算环节能更加合理地描述载体的刚体空间运动,四元数法的研究得到了广泛重视。

1. 四元数算法

四元数法是捷联惯导中的一种算法[5],利用陀螺仪、加速度计(有条件的时候可以加上磁强计)的数据,将四轴机体坐标系与地球坐标系进行耦合,最后计算出正确的四轴姿态方位数据。

四元数的数学概念是 1843 年由哈密顿首先提出的,它是代数学中的内容之一。随着捷联式惯性导航技术的发展,为了更简便地描述刚体的角运动,采用了四元数这个数学工具,用它来弥补通常描述刚体角运动的 3 个欧拉角参数在设计控制系统时的不足。四元数可以描述一个坐标系或一个矢量相对某一个坐标系的旋转,四元数的标量部分表示了转角的一半余弦值,而其矢量部分则表示瞬时转轴的方向、瞬时转动轴与参考坐标系轴间的方向余弦值。因此,一个四元数既表示了转轴的方向,又表示了转角的大小,往

往称为转动四元数。

工程上一般运用范数为 1 的特征四元数，特征四元数的标量部分表示转角的一般余弦值，其矢量部分表示瞬时转轴 n 的方向。矢量 $\boldsymbol{R}$ 相对参考坐标系旋转一个转角 θ，旋转轴 n 的方向由四元数的虚部确定，$\cos\alpha$、$\cos\beta$、$\cos\gamma$ 表示旋转轴 n 与参考坐标系轴间的方向余弦值。

$$\boldsymbol{R}' = q\boldsymbol{R}q' \tag{2.2.3}$$

式中，$\boldsymbol{R}$ 为某矢量；$q = p_0 + p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k}$，其中

$$p_0 = \cos\left(\frac{\theta}{2}\right)$$

$$p_1 = \sin\left(\frac{\theta}{2}\right)\cos\alpha$$

$$p_2 = \sin\left(\frac{\theta}{2}\right)\cos\beta$$

$$p_3 = \sin\left(\frac{\theta}{2}\right)\cos\gamma$$

四元数姿态矩阵微分方程式只要解 4 个一阶微分方程式组即可，比方向余弦姿态矩阵微分方程式计算量有明显的减少，能满足工程实践中对实时性的要求。

2. 四元数转欧拉角算法流程

运用机体上搭载的加速度计与陀螺的数据求出初始欧拉角。为了获取四旋翼飞行器当前准确、可靠的姿态信息（俯仰角、横滚角、偏航角），用于四轴的平衡及运动控制，应当首先进行数据融合。

1）单独使用加速度计测量姿态时

加速度计测量对象为比力，理想情况下想要得到的是重力加速度分量，但是任何机械系统不可避免存在的平动振动也会产生相当大的加速度，加速度计特别容易受飞行器机架振动的影响，振动振幅高达 2 个重力单位。而加速度计完全无法区分重力加速度和外力加速度这两种加速度。所以加速度计虽然能够精确测量出当前飞行器的各个加速度分量，且测量值随时间的变化相对较小，但是飞行器的线性振动给测量引入了强烈的噪声，使得这个数据不能直接使用。

2）单独使用陀螺仪测量姿态时

陀螺仪对振动不敏感，受外部影响弱，稳定性好，但输出量为角速度，需要积分才能得到姿态。短期内这个姿态数据具有很大的参考价值，但是由于陀螺仪器件本身误差的存

在，会导致这个姿态数据随着长时间的积分逐渐偏离正确值，即测量值随时间变化比较大。

因此，可以看出，将这两种传感器的优缺点互补，将两者的数据融合起来才能得到更好的效果。

数据融合的方法有很多种，Madgwick 的 IMUupdate 算法就是其中之一。

IMUupdate 算法中要求输入 6 个数据，$\boldsymbol{g}_x$、$\boldsymbol{g}_y$、$\boldsymbol{g}_z$分别为陀螺仪测量到的 3 个旋转角速度分量，$\boldsymbol{a}_x$、$\boldsymbol{a}_y$、$\boldsymbol{a}_z$则为加速度计的 3 个加速度分量，IMUupdate 函数体如下：

```
void IMUupdate(float gx, float gy, float gz, float ax, float ay, float az)
```

首先，将加速度计测量到的 3 个向量转换为单位向量，代码如下：

```
norm = sqrt(ax * ax + ay * ay + az * az);
ax = ax / norm;
ay = ay / norm;
az = az / norm;
```

根据余弦矩阵和欧拉角的定义，将地理坐标系的重力向量转到机体坐标系，得到的$\boldsymbol{v}_x$、$\boldsymbol{v}_y$、$\boldsymbol{v}_z$，其实就是在当前的四元数机体坐标参照系上换算出来的重力单位向量，即

$$\boldsymbol{v}_x = 2(q_1 q_3 - q_0 q_2)$$
$$\boldsymbol{v}_y = 2(q_0 q_1 + q_2 q_3)$$
$$\boldsymbol{v}_z = q_0 q_0 - q_1 q_1 - q_2 q_2 + q_3 q_3$$

$\boldsymbol{a}_x$、$\boldsymbol{a}_y$、$\boldsymbol{a}_z$是机体坐标参照系上加速度计测出来的重力向量，也就是实际测出来的重力向量。$\boldsymbol{v}_x$、$\boldsymbol{v}_y$、$\boldsymbol{v}_z$为用陀螺积分后的姿态来推算出的重力向量，它们都是机体坐标参照系上的重力向量。那它们之间的误差向量，就是陀螺积分后的姿态和加速度计测出来的姿态之间的误差。向量间的误差，可以用向量叉积（也叫向量外积、叉乘）来表示，$\boldsymbol{e}_x$、$\boldsymbol{e}_y$、$\boldsymbol{e}_z$就是两个重力向量的叉积，即

$$\boldsymbol{e}_x = (\boldsymbol{a}_y \boldsymbol{v}_z - \boldsymbol{a}_z \boldsymbol{v}_y)$$
$$\boldsymbol{e}_y = (\boldsymbol{a}_z \boldsymbol{v}_x - \boldsymbol{a}_x \boldsymbol{v}_z)$$
$$\boldsymbol{e}_z = (\boldsymbol{a}_x \boldsymbol{v}_y - \boldsymbol{a}_y \boldsymbol{v}_x)$$

这个向量叉积仍然是位于机体坐标系上的，而陀螺积分误差也是在机体坐标系上，而且叉积的大小与陀螺积分误差成正比，正好拿来纠正陀螺。由于陀螺是对机体直接积分，所以对陀螺的纠正量会直接体现在对机体坐标系的纠正。用叉积误差 $\boldsymbol{e}_x$、$\boldsymbol{e}_y$、$\boldsymbol{e}_z$来做 PI 调节，最后用得到的调节量对陀螺仪进行零偏修正，代码如下：

```
//integral error scaled integral gain
exInt = exInt + ex * Ki;
eyInt = eyInt + ey * Ki;
ezInt = ezInt + ez * Ki;
//adjusted gyroscope measurements
gx = gx + Kp * ex + exInt;
gy = gy + Kp * ey + eyInt;
gz = gz + Kp * ez + ezInt;
```

其中定义了 Kp 和 Ki 的值为

```
#define Kp 1.6f
#define Ki 0.001f
```

在上一步已经得到了修正过零偏的陀螺仪数据 $\boldsymbol{g}_x$、$\boldsymbol{g}_y$、$\boldsymbol{g}_z$，在这里设机体坐标系相对平台坐标系的转动四元素为

$$Q = q_0 + q_1 i_h + q_2 j_h + q_3 k_h$$

然后使用一阶龙格—库塔法对陀螺仪角速度值进行处理更新得到新的四元数值。一阶龙格—库塔法计算式为

$$q(t+T) = q(t) + T\Omega_h(t)q(t)$$

将角速度值代入上式展开，得到

$$q_0(t+T) = q_0(t) + \frac{T}{2}[-w_x(t)q_1(t) - w_y q_2(t) - w_z q_3(t)]$$

$$q_1(t+T) = q_1(t) + \frac{T}{2}[w_x(t)q_0(t) - w_z q_2(t) - w_y q_3(t)]$$

$$q_2(t+T) = q_2(t) + \frac{T}{2}[w_y(t)q_0(t) - w_z q_1(t) + w_x q_3(t)]$$

$$q_3(t+T) = q_3(t) + \frac{T}{2}[w_z(t)q_0(t) + w_y q_1(t) - w_x q_2(t)]$$

具体的实现代码如下：

```
#define halfT 0.001f
q0 = q0 + (-q1 * gx - q2 * gy - q3 * gz) * halfT;
q1 = q1 + (q0 * gx + q2 * gz - q3 * gy) * halfT;
q2 = q2 + (q0 * gy - q1 * gz + q3 * gx) * halfT;
q3 = q3 + (q0 * gz + q1 * gy - q2 * gx) * halfT;
```

其中定义了 halfT 的值为 0.001，代表四元数固定为 2ms 更新一次数据，halfT 取值为周期的一半，即 $T/2$。

在上面的四元数微分方程更新了四元数数据后，还需要将得到的四元数进行规范化

处理，规范化的公式及过程代码如下：

$$q_i = \frac{q_i}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}} \tag{2.2.4}$$

```
norm = sqrt( q0 * q0 + q1 * q1 + q2 * q2 + q3 * q3)
q0 = q0/norm
q1 = q1/norm
q2 = q2/norm
q3 = q3/norm
```

至此已经获取到了稳定、有效的姿态四元数数据，但由于在设计控制规律时，欧拉角的直观性和几何意义仍然采用欧拉角描述，因此还需要将四元数转换为欧拉角，即

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{bmatrix} \arctan \dfrac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \arctan(2(q_0q_2 - q_1q_3)) \\ \arctan \dfrac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix} \tag{2.2.5}$$

arctan 的结果是，这并不能够覆盖所有的角度(对于角的取值范围已经满足)，因此需要用 arctan2 来代替 arctan，修正后的公式为

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{bmatrix} \operatorname{arctan2}(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ \arcsin(2(q_0q_2 - q_1q_3)) \\ \operatorname{arctan2}(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix} \tag{2.2.6}$$

实际 C 语言代码如下：

```
angle->pit = asin(-2 * q1 * q3 + 2 * q0 * q2) * 57.3;
angle->rol = -atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2 * q2 + 1) *
57.3;
angle->yaw = atan2( 2 * (q0 * q3 + q1 * q2), 1 - 2 * q2 * q2 - 2 * q3 * q3 ) * 57.3;
```

至此姿态解算得到四旋翼飞行器稳定、可靠的姿态角度，可以用于四轴的飞行稳定控制中。

2.3 平衡控制算法

2.3.1 四旋翼飞行器的 PID 调节原理

飞行控制部分解码出遥控信号或者采集到姿态传感器数据之后，随即将两者进行融

合，根据融合的结果对姿态进行相应调整，即通过 PWM 信号对电机进行控制，这时就需要设置合适的 PID 控制率来输出合适的控制量，以避免电机转速大幅度的变化而引起飞行器剧烈抖动。

工业控制领域将这种把偏差的比例、积分和微分通过线性组合构成控制量，用这一控制量对被控制对象进行控制的方法称为 PID 控制[6]，原理如图 2.3.1 所示。

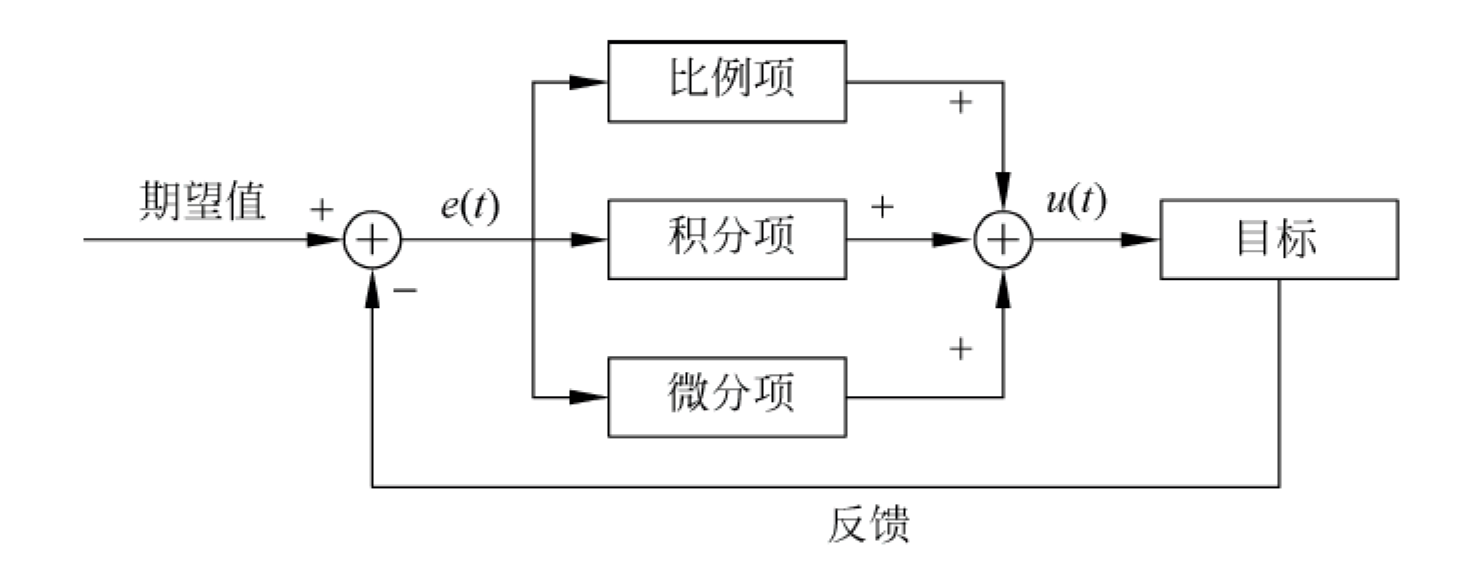


图 2.3.1　PID 控制原理框图

PID 控制的优势在于结构简单、数学原理清晰、易于实现且性能优良，常用的公式为

$$\begin{cases} u(t) = K_P e(t) + K_I \int_o^t e(\tau)\mathrm{d}\tau + K_D\left(\dfrac{\mathrm{d}e(t)}{\mathrm{d}t}\right) \\ e(t) = x_d(t) - x(t) \end{cases} \tag{2.3.1}$$

式中，$e(t)$为期望值与当前状态值的偏差；$u(t)$为 PID 调节后的系统控制输入量；K_P、K_I和 K_D分别为 PID 控制器的比例系数、积分系数和微分系数。简单来说，PID 控制器各校正环节的作用如下。

1. 比例环节

比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。偏差一旦产生，控制器立即发生作用调节控制输出，使被控量朝着减小偏差的方向变化，偏差减小的速度取决于比例系数 K_P，K_P越大则偏差减小得越快。

2. 积分环节

在积分控制中，控制器的输出与输入误差信号的积分成正比关系。对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误差的或简称有差系统。为了消除稳态误差，在控制器中必须引入“积分项”。积分项误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。

3. 微分环节

在微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在有较大惯性组件（环节）或有滞后组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。

由于在单片机数字系统中，单片机控制是一种采样控制，它只能根据采样时刻的偏差值计算控制量，所以使用的是数字 PID 控制器，数字 PID 控制算法通常又分为位置式 PID 控制算法和增量式 PID 控制算法。

1）位置式 PID 控制算法

对原始 PID 控制函数进行离散化处理，对于模拟 PID 控制算法，用一系列的采样时刻点 kT 代表连续的时间 t，以和式代表积分，以增量代表微分，则可得到

$$\left\{\begin{array}{l} t = kT \quad (k = 0,1,2,\cdots) \\ \int_0^t e(t)\mathrm{d}t \approx T\sum_{j=0}^{k} e(jT) = T\sum_{j=0}^{k} e(j) \\ \dfrac{\mathrm{d}e(t)}{\mathrm{d}t} \approx \dfrac{e(kT)-e[(k-l)T]}{T} = \dfrac{e(k)-e(k-l)}{T} \end{array}\right\} \tag{2.3.2}$$

式中，T 为采样周期。当然，在上述离散化过程中，采样周期必须足够短才能保证有足够的精度。为了使式子看起来简洁，将 $e(kT)$ 简化表示为 $e(k)$，可得到离散的 PID 表达式为

$$u(k) = K_{\mathrm{P}}\left[e(k) + \frac{T}{T_i}\sum_{j=0}^{k} e(j) + \frac{T_{\mathrm{d}}}{T}(e(k)-e(k-1))\right] \tag{2.3.3}$$

将上式展开，得到的三项分别为比例项、积分项和微分项。由于单片机输出的 $u(k)$ 直接去控制执行机构（如阀门），$u(k)$ 的值和执行机构的位置（如阀门的打开程度）是一一对应的，所以通常称该式为位置式 PID 控制算法，其流程如图 2.3.2 所示。

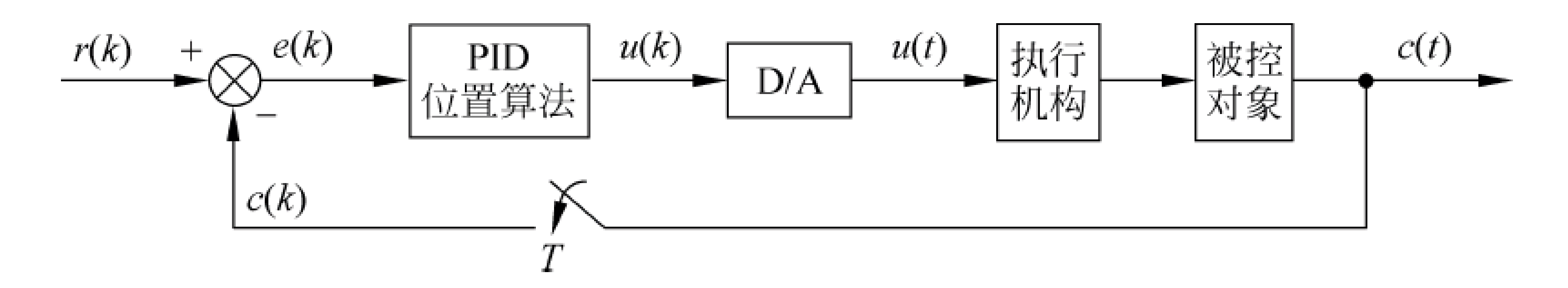


图 2.3.2　位置式 PID 控制系统

这种控制算法的缺点是：由于是全量输出，所以每次输出都和过去的状态有关，计算时会对 $e(k)$ 进行累加，工作量较大；单片机输出的 $u(k)$ 对应的是执行机构的实际位置，如果计算出现故障，则输出的 $u(k)$ 会产生大幅度变化，引起执行机构的大幅度变化。在实际应用中，一些安全性要求较高的系统必然要避免出现这种现象，因此产生了增量式 PID 控制算法，即数字控制器的输出只是控制量的增量。

2）增量式 PID 控制算法

由式(2.3.3)，根据递推原理可得

$$u(k-1)=K_{\mathrm{P}}\left[e(k-1)+\frac{T}{T_i}\sum_{j=0}^{k-1}e(j)+\frac{T_{\mathrm{d}}}{T}(e(k-1)-e(k-2))\right] \quad (2.3.4)$$

将式(2.2.3)减去式(2.2.4)，得

$$\Delta u(k)=u(k)-u(k-1)=Ae(k)+Be(k-1)+Ce(k-2)$$

$$A=K_{\mathrm{P}}\left(1+\frac{T}{T_i}+\frac{T_{\mathrm{d}}}{T}\right)$$

$$B=K_{\mathrm{P}}\left(1+\frac{2T_{\mathrm{d}}}{T}\right)$$

$$C=K_{\mathrm{P}}T_{\mathrm{d}}/T$$

A、B、C 都是与采样周期、比例系数、积分时间常数、微分时间常数有关的系数。可以看出，只要使用前后 3 次测量值的偏差即可由上式求出控制增量，流程图如 2.3.3 所示。

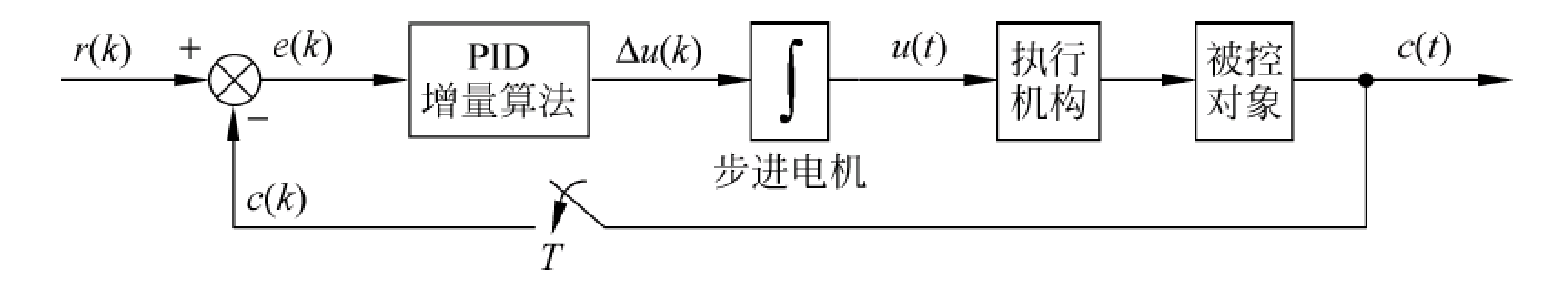


图 2.3.3　增量式 PID 控制系统

就整个系统而言，位置式与增量式 PID 控制算法并无本质区别，增量式控制虽然只是算法上做了一点改进，却带来了不少优点。

① 由于单片机输出增量，所以误动作时影响小，必要时可以用逻辑判断的方法去掉。

② 手动/自动控制切换时冲击小，以便于实现无扰动切换。

③ 算式中不需要累加运算，控制增量的确定仅与最近 k 次的采样值有关，所以较容易通过加权处理获得比较好的控制效果。

在后面的各个控制环节中根据其控制环节的控制特性的不同选用合适的 PID 控制算法。对于一般四旋翼飞行器而言，有 6 个状态检测量(3 个空间坐标值和 3 个角度值)，

但是只有 4 个控制输出量(4 个电机转速)。总体升力 T 决定四旋翼飞行器 Z 轴方向的加速度,从而保证飞行器保持在空中,扭力τ_B则决定着 3 个姿态角的角加速度,因此对于四旋翼飞行器仿真,此处也可以采用简易版的 PD 控制器,即

$$T=(g+K_{z,\mathrm{D}}(\dot{z}_{\mathrm{d}}-\dot{z})+K_{z,\mathrm{P}}(z_d-z))\left(\frac{m}{C_\phi C_\theta}\right)$$

$$\tau_\phi=(K_{\phi,\mathrm{D}}(\dot{\phi}_d-\dot{\phi})+K_{\phi,\mathrm{P}}(\phi_{\mathrm{d}}-\phi))I_{xx}$$

$$\tau_\theta=K_{\theta,\mathrm{D}}(\dot{\theta}_{\mathrm{d}}-\dot{\theta})+K_{\phi,\mathrm{P}}(\theta_{\mathrm{d}}-\theta))\ I_{yy}$$

$$\tau_\psi=(K_{\psi,\mathrm{D}}(\dot{\psi}_{\mathrm{d}}-\dot{\psi})+K_{\phi,\mathrm{P}}(\psi_{\mathrm{d}}-\psi))I_{zz}$$

结合电机的转速,便能计算出达到期望状态所需的电机控制量,仿真用电机转速表示为

$$\omega_1^2=\frac{T}{4k}-\frac{\tau_\theta}{2kl}-\frac{\tau_\psi}{4b}$$

$$\omega_2^2=\frac{T}{4k}-\frac{\tau_\phi}{2kl}+\frac{\tau_\psi}{4b}$$

$$\omega_3^2=\frac{T}{4k}+\frac{\tau_\theta}{2kl}-\frac{\tau_\psi}{4b}$$

$$\omega_4^2=\frac{T}{4k}+\frac{\tau_\phi}{2kl}+\frac{\tau_\psi}{4b}$$

PID 计算的程序如下:

```
double PIDCalc( PID *pp, double NextPoint )
{
    double dError, Error;
    Error = pp->SetPoint - NextPoint;           //偏差
    pp->SumError += Error;                      //积分
    dError = pp->LastError - pp->PrevError;     //当前微分
    pp->PrevError = pp->LastError;
    pp->LastError = Error;
    return (pp->Proportion * Error              //比例项
     + pp->Integral * pp->SumError              //积分项
     + pp->Derivative * dError                  //微分项
    );
}
```

PID 程序设计流程图如 2.3.4 所示。

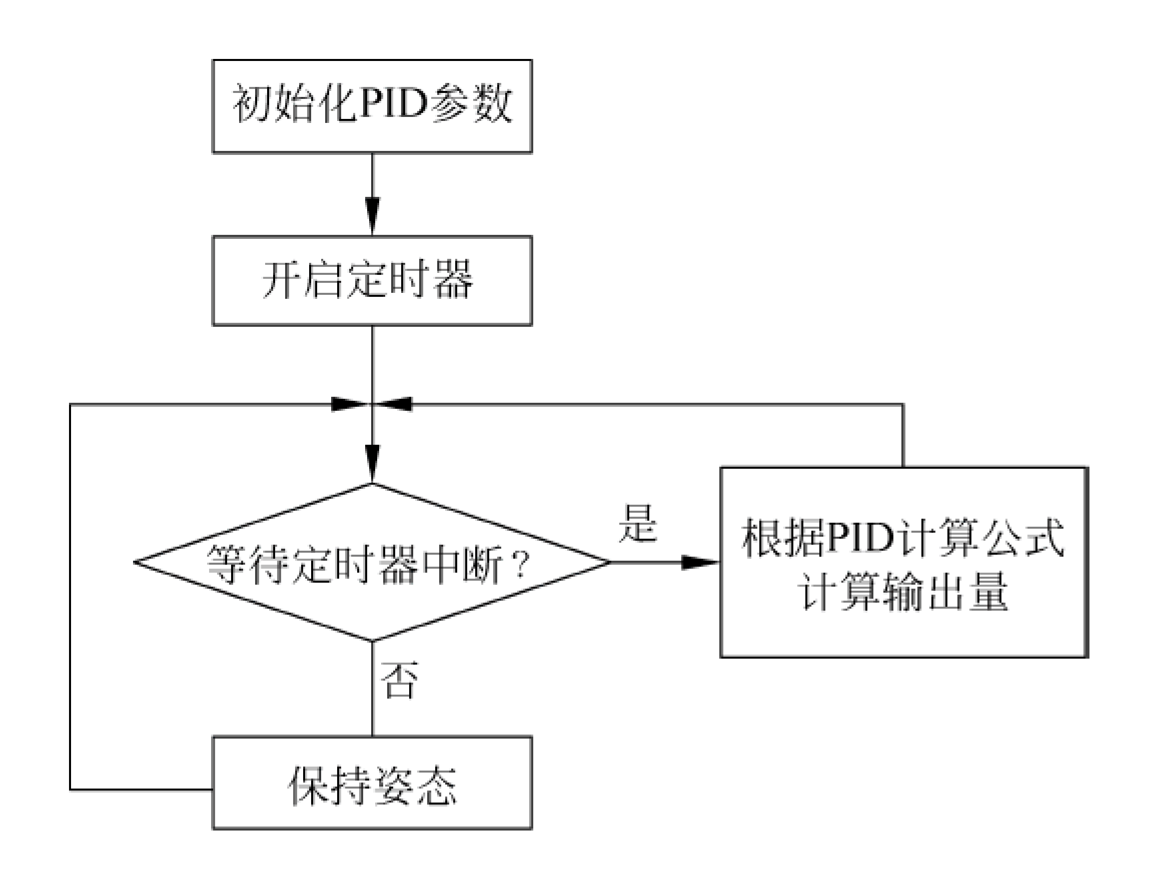


图 2.3.4　PID 程序设计流程图

2.3.2　四旋翼飞行器的 PID 参数整定(利用 MATLAB 仿真确定理论参数)

PID 算法能让被控系统表现出优良的性能，但这在很大程度上取决于所选取的 3 个参数。选取 3 个参数值，在实际系统中受到一些条件约束，常常凭借经验通过人工测试选取，这既费时又费力，而且还不能确保所选取的值是否为最优。在仿真系统中，试验可以被任意次地重复，从而给自动选取最优 PID 参数提供了可能，仿真系统与实际系统越接近，则选取的 PID 参数越具有参考价值。自动选取最优 PID 参数的算法有很多，如内部模型控制、遗传算法(GA)和进化计算技术，此处选用简化版的极值搜索算法。

该算法思想很简单，即找到一组最优参数$\boldsymbol{\theta}=(K_{\mathrm{P}},K_{\mathrm{I}},K_{\mathrm{D}})$，使得设定的代价函数

$$J(\boldsymbol{\theta})=\frac{1}{t_{\mathrm{f}}-t_0}\int_{t_0}^{t_{\mathrm{f}}} e(t,\theta)^2\,\mathrm{d}t \tag{2.3.5}$$

值最小。此代价函数用来对误差进行评估，单位时间内偏差越大则该值越大，其中的$e(t,\boldsymbol{\theta})$表示在设定参数为$\boldsymbol{\theta}$时，在有初始干扰的情况下，实际轨迹(坐标值集合)与参考值之间的偏差。假如可以计算代价函数$J(\boldsymbol{\theta})$的梯度$\nabla J(\boldsymbol{\theta})$，那么只要按照梯度下降算法就可以不断地改善参数值，即

$$\boldsymbol{\theta}(k+1)=\boldsymbol{\theta}(k)-\alpha\nabla J(\boldsymbol{\theta}) \tag{2.3.6}$$

式中，$\boldsymbol{\theta}(k)$为 k 次迭代后的参数值；α 为调节系数，表示每次调节的力度，当 $k\to\infty$时，代价函数$J(\boldsymbol{\theta})$达到局部最优值。$J(\boldsymbol{\theta})$的定义式为

$$\nabla J(\boldsymbol{\theta})=\left(\frac{\partial}{\partial K_{\mathrm{P}}}J(\boldsymbol{\theta}),\frac{\partial}{\partial K_{\mathrm{I}}}J(\boldsymbol{\theta}),\frac{\partial}{\partial K_{\mathrm{D}}}J(\boldsymbol{\theta})\right) \tag{2.3.7}$$

根据 $J(\boldsymbol{\theta})$ 可以近似计算它的偏导，即

$$\frac{\partial}{\partial K}J(\boldsymbol{\theta})\approx\frac{J(\boldsymbol{\theta}+\delta\cdot\boldsymbol{u}_k)-J(\boldsymbol{\theta}-\delta\cdot\boldsymbol{u}_k)}{2\delta} \tag{2.3.8}$$

式中，$\boldsymbol{u}_k$ 为方向的单位向量，当 $\delta\rightarrow 0$ 时这种近似更加精确。至此，根据式(2.3.5)～式(2.3.8)，便可以计算代价函数的最小值和局部最优参数组。算法流程如图 2.3.5 所示。

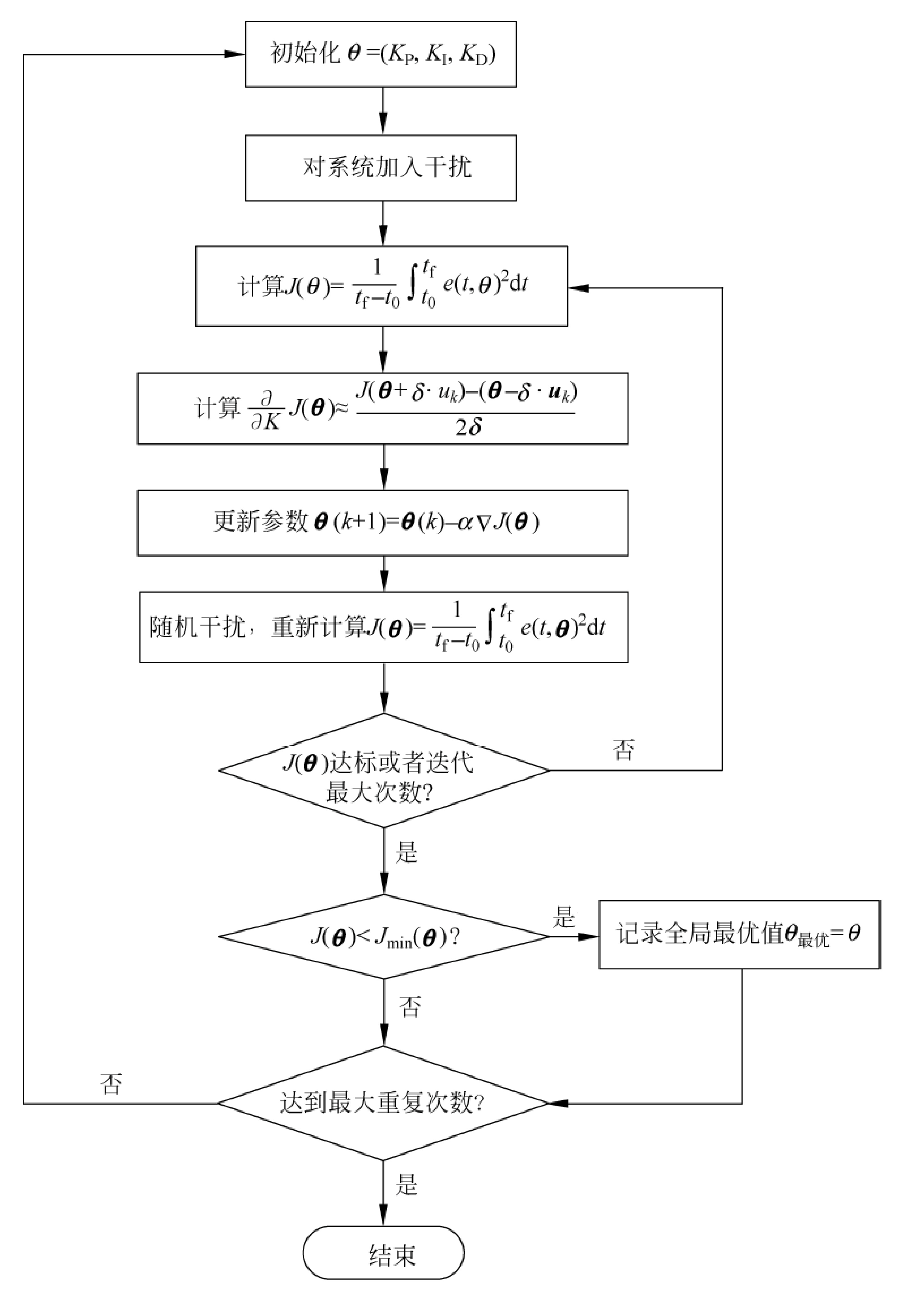


图 2.3.5 PID 参数自动选取算法

2.4 滤波算法

2.4.1 卡尔曼滤法算法

由于加速度计本身的特性,以及飞行器 4 个电机和机体的振动,安装在飞行器上的三轴加速度计采集的 3 个轴的角速度 ω 数据含大量的噪声,而陀螺仪本身受噪声影响较小,因此必须要采取一种算法,对加速度计的数据进行滤波,去除噪声,再和陀螺仪的数据进行融合,以消除陀螺仪的累积误差,从而得到飞行器正确的姿态。一种理想的算法应具有以下特性。

① 积分数值漂移小。在离散系统中,常用数据的累加代表积分。但如果对采集到的陀螺仪数据直接积分,得到的数据并不稳定,无法真实地反映姿态数据,因此需要用加速度计补偿陀螺,使积分稳定下来。

② 能较好地处理较小的积分。某一个轴向的积分越多,则 Ail_cal/Ele_cal 等越大,电机的反应也会越大。而如果只倾斜了很小角度,则电机没有足够的力量来拉平这个小角度;如果融合后的积分在小角度没有拉平时一直变大,则电机有足够的力量拉平小角度。

③ 积分数值不溢出。积分数值相对于电机数值应该不能太大,以便计算k_P、k_I、k_D参数。由于积分是始终不清零的,因此用于反映姿态数据的关键参数时还需要做好数据类型的设计与限幅,以防止溢出。

④ 长期稳定性。算法应该尽量健壮有效,程序不会跑飞,即使工作了很久,不论是零点积分还是晃动之后的调整反应,都应该是迅速有效的。

⑤ 与后期 PID 算法的配合。

卡尔曼算法是一种“最优化自回归数据处理算法(Optimal Recursive Data Processing Algorithm)”,对于解决很大部分的问题,它是最优、效率最高的算法[7]。1960 年,匈牙利数学家鲁道夫·卡尔曼提出了卡尔曼滤波的算法。卡尔曼滤波的思路并不是得出完全正确无误的当前状态,而是采用一种递归自更新的方法来估计出当前的状态,且保证原状态与估计出来的状态的均方误差收敛至最小。卡尔曼算法的广泛应用已经超过50 年,其应用领域包括机器人导航、控制、传感器数据融合甚至在军事方面的雷达系统以及导弹追踪等。

一般测得的信号都是受到噪声干扰的，如加速度计所测的信号。怎样有效地抑制或消除噪声，将所需信号完整提取出来，是信号处理的目的。滤除噪声的系统称为滤波器。先只考虑加性噪声的影响，观测数据 $x(n)$ 是信号 $s(n)$ 与噪声 $v(n)$ 之和（如图 2.4.1 所示），即

$$x(n) = s(n) + v(n)$$

一般的信号处理模型如图 2.4.2 所示。

x(n)=s(n)+v(n)
s(n)
x(n)
v(n)

图 2.4.1 观测数据、信号与噪声的关系

x(n)
s(n)+v(n)
h(n)
y(n)

图 2.4.2 一般信号处理模型

假设信号和噪声没有相关性（即互相关为 0），则有

$$H_{\text{opt}}(z) = \frac{S_{xs}(z)}{S_{xx}(z)} = \frac{S_{ss}(z)}{S_{ss}(z) + S_{vv}(z)} \tag{2.4.1}$$

由此可见：没有噪声时，信号得到还原；没有输入信号时，噪声完全被滤除。维纳滤波能够随着信号的强弱自适应地将噪声抑制掉。

维纳滤波的依据是全部的过去测量值和当前状态 k 时刻下的测量值进行比对，以这二者误差的均方值收敛到某个最小值为目标。维纳滤波的结果是一个确定的传递函数 H。卡尔曼滤波所需的样本依据只是当前状态 k 时刻的估计值和前一个状态 $k-1$ 时刻的测量值通过不断递归更新的方法，并不需要追溯过去所有的观测值，但是目标同样是误差的均方值收敛到最小值。卡尔曼滤波的结果是预估的状态值。卡尔曼滤波具有以下特点。

① 采用递归的算法。递归的算法就是将 k 时刻卡尔曼滤波计算得出的状态值作为 $k+1$ 时刻的测量值。

② 用递归法计算并不需要知道全部过去的值，用状态方程描述状态变量的动态变化规律，因此信号可以是平稳的，也可以是非平稳的。卡尔曼滤波适用于平稳和非平稳过程。从这一点可以看出，采用卡尔曼滤波算法控制姿态的飞行器，其初始状态可以不是稳态。

③ 卡尔曼滤波采取的误差准则为估计误差的均方值最小。假设某系统 k 时刻的状态变量为 x_k，则状态方程和量测方程，也就是输出方程，表示为

$$x_{k+l} = \boldsymbol{A}_k x_k + \omega_k \tag{2.4.2}$$

$$y_k = \boldsymbol{C}_k x_k + v_k \tag{2.4.3}$$

式中，$\boldsymbol{A}$ 为时间，指系统的第 $\boldsymbol{A}$ 时刻相应时刻信号的取值；x_{k+l} 为 $k+l$ 时刻的状态变量；y_k 为 k 时刻的输出信号；$\boldsymbol{C}_k$ 为 k 时刻状态变量 x_k 与输出信号之间的增益矩阵。卡尔曼滤波信号处理模型如图 2.4.3 所示。

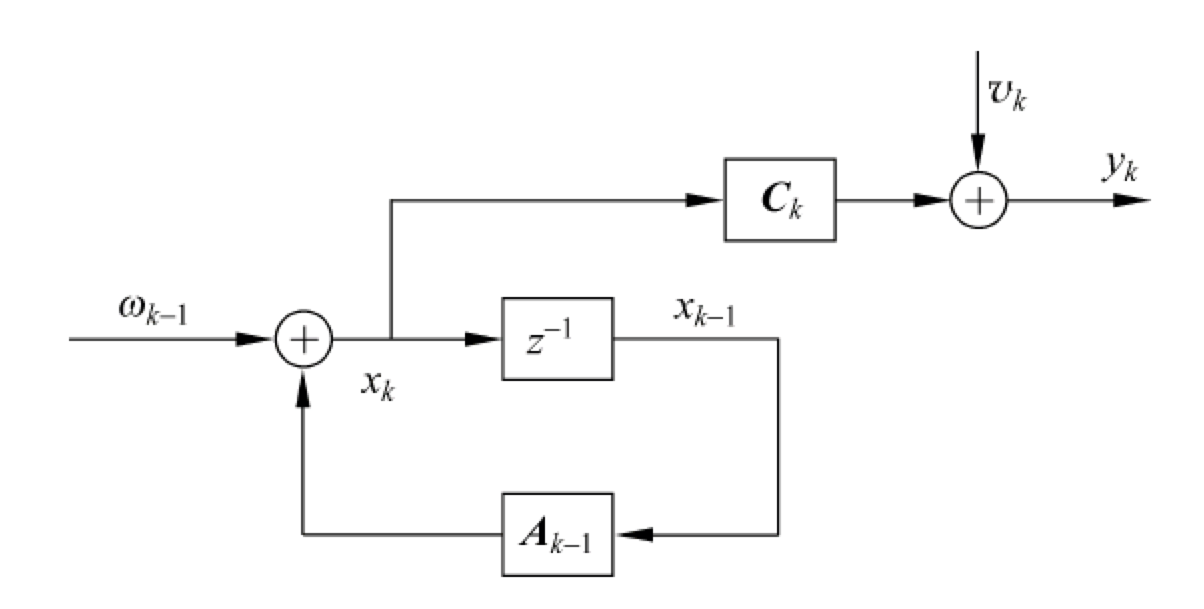


图 2.4.3　卡尔曼滤波信号处理模型

将状态方程中时间变量 k 用 $k-l$ 代替，得到的状态方程为

$$x_k = \boldsymbol{A}_k x_{k-l} + \omega_{k-l} \tag{2.4.4}$$

那么输出方程为

$$y_k = \boldsymbol{C}_k x_k + v_k \tag{2.4.5}$$

基于一些理想的假设，有

$$\omega_k : E[\omega_k] = 0, \quad \sigma_\omega^2 = Q_k, \quad \gamma_{\omega_k, \omega_j} = Q_k \delta_{kj}$$

$$v_k : E[v_k] = 0, \quad \sigma_v^2 = R_k, \quad \gamma_{v_k, v_j} = R_k \delta_{kj}$$

$$\delta_{kj} = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$$

其中，先忽略系统中测得的噪声和输入系统的信号，假设这二者为零。卡尔曼滤波递推回归的思路是这样的：先得估计状态变量以及估计输出信号，得到这二者的估计值。目标是让状态变量的预估与实际状态变量测量值的差值即均方差收敛以达到最小，就必须用输出信号的估计值通过某种加权来修正状态变量的估计值。因此，卡尔曼滤波算法实现的基础是得到使均方误差达到最小的加权矩阵的最优值。

现在先假设输入信号和观测噪声为零，此时的状态方程为

$$\hat{x}'_k = \boldsymbol{A}_k \hat{x}_{k-1} \tag{2.4.6}$$

量测方程为

$$\hat{y}'_k = \boldsymbol{C}_k \hat{x}'_k = \boldsymbol{C}_k \boldsymbol{A}_k \hat{x}_{k-1} \tag{2.4.7}$$

式中，$\boldsymbol{A}_k$ 为增益参数矩阵；$\boldsymbol{C}_k$ 为校正参数矩阵。

假设噪声为 0，有一个误差存在于输出信号的估计值与实测值之间，这个误差值用$\tilde{y}_k$表示，那么就有

$$\tilde{y}_k = y_k - \hat{y}'_k \tag{2.4.8}$$

根据卡尔曼滤波的递推思想，为了提高状态估计的准确度，用输出信号的估计误差来校正状态变量。于是就得到

$$\begin{aligned}\hat{x}_k &= \boldsymbol{A}_K \hat{x}_{k-1} + \boldsymbol{H}_k (y_k - \hat{y}'_k) \\ &= \boldsymbol{A}_K \hat{x}_{k-1} + \boldsymbol{H}_k (y_k - \boldsymbol{C}_k \boldsymbol{A}_k \hat{x}_{k-1})\end{aligned} \tag{2.4.9}$$

式中，$\boldsymbol{H}_k$ 为一个加权矩阵。

经过校正后的状态变量的估计误差及其均方值分别用$\tilde{x}_k$和 $\boldsymbol{P}_k$ 来表示。把未经校正的状态变量的估计误差的均方值用 $\boldsymbol{P}'_k$ 来表示。校正后的状态变量为

$$\tilde{x}_k = x_k - \hat{x}_k \tag{2.4.10}$$

即状态变量减去不考虑噪声时的状态变量，那么均方值为

$$\boldsymbol{P}'_k = E[(x_k - \hat{x}'_k)(x_k - \hat{x}'_k)^{\mathrm{T}}] \tag{2.4.11}$$

未经过校正的均方值为

$$\boldsymbol{P}_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^{\mathrm{T}}] \tag{2.4.12}$$

按照卡尔曼滤波算法，状态变量的估计值与实际值的误差的均方值要达到最小，首先要推导出状态变量的估计值和估计误差之间的关系式，再调整 $\boldsymbol{H}_k$ 使 $\boldsymbol{P}_k$ 达到最小。

那么要得到卡尔曼滤波的递推公式，先推导状态变量的估计值和状态变量的估计误差，然后再计算均方值 $\boldsymbol{P}_k$，之后再通过化简得到卡尔曼滤波算法 4 个递推公式，即

$$\begin{cases}\hat{x}_k = \boldsymbol{A}_k \hat{x}_{k-1} + \boldsymbol{H}_k (y_k - \boldsymbol{C}_k \boldsymbol{A}_k \hat{x}_{k-1}) \\ \boldsymbol{H}_k = \boldsymbol{P}'_k \boldsymbol{C}_k^{\mathrm{T}} (\boldsymbol{C}_k \boldsymbol{P}'_k \boldsymbol{C}_k^{\mathrm{T}} + \boldsymbol{R}_k)^{-1} \\ \boldsymbol{P}'_k = \boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A}_k^{\mathrm{T}} + \boldsymbol{Q}_{k-1} \\ \boldsymbol{P}_k = (I - \boldsymbol{H}_k \boldsymbol{C}_k) \boldsymbol{P}'_k\end{cases} \tag{2.4.13}$$

卡尔曼滤波算法的递推流程如下。

① 从一个时刻的状态变量的估计值$\hat{x}_{k-1}$和它的均方值 $\boldsymbol{P}_{k-1}$ 导出未更新的均方值：$\boldsymbol{P}'_k = \boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A}_k^{\mathrm{T}} + \boldsymbol{Q}_{k_1}$。

② 得到的 $\boldsymbol{P}'_k$代入 $\boldsymbol{H}_k = \boldsymbol{P}'_k \boldsymbol{C}_k^{\mathrm{T}} (\boldsymbol{C}_k \boldsymbol{P}'_k \boldsymbol{C}_k^{\mathrm{T}} + \boldsymbol{R}_k)^{-1}$ 中，得到加权参数矩阵 $\boldsymbol{H}_k$。

③ 将 $\boldsymbol{H}_k$ 代入 $\hat{x}_k=\boldsymbol{A}_k\hat{x}_{k-1}+\boldsymbol{H}_k(y_k-\boldsymbol{C}_k\boldsymbol{A}_k\hat{x}_{k-1})$ 得到 $\hat{x}_k$。

④ 将 $\boldsymbol{H}_k$ 代入 $\boldsymbol{P}_k=(\boldsymbol{I}-\boldsymbol{H}_k\boldsymbol{C}_k)\boldsymbol{P}_k'$，得到 $\boldsymbol{P}_k'$。

至此，完成了卡尔曼滤波的一次完整的递推过程。卡尔曼滤波是不断递归的运算过程，因此在完成前一次的递推过程后，就变为下一次递推运算的输入。由此可以看出，若起始状态是非稳态，则卡尔曼滤波需要一定的时间进行若干次递归运算来达到稳态的收敛的输出值。

2.4.2 互补滤波算法

互补滤波算法的思路是，加速度计由于其自身灵敏度较高。在振动较大的场合，加速度计的测量值容易混入振动等噪声[8]。也就是说，加速度计在瞬时精度上容易受到外部干扰而有很大误差，直接用加速度计瞬时值通过三角函数计算得出的角度误差很大。陀螺仪的测量值是角速度。短时间内将角速度按时间积分得到的角度是比较准确的。但是此时得到的角度带有累积误差，可以利用这两种传感器的特性来互补得到正确的角度。实现互补滤波的思路就是将加速度计测量值做一个均值，通过三角函数计算出角度，利用这个值来视时间长短加权校正陀螺仪测得并换算的角度值，达到去除陀螺仪角度累积误差的效果。简单来说，就是短时间内陀螺仪的测量值比较准确，以它为主来计算得到姿态；长时间上看用加速度计滤波后的测量值比较准确，这时候加大加速度计纠正偏差的加权比例。

互补滤波算法就是把加速度计算出的角度取均值，构造成角度上的校正量，叠加到陀螺仪测出的角度上，实现数据融合。实现过程是将加速度计的数据进行均值运算。均值后的加速度计值为

$$a_{cc}=\frac{1}{n}\sum_{n}A_{cc} \tag{2.4.14}$$

式中，a_{cc} 为均值后的加速度计值。设 g 为重力加速度，先用 a_{cc} 换算出角度 $\mathrm{A_{CC}}$，即

$$\mathrm{A_{CC}}=\arccos\left(\frac{a_{cc}}{g}\right) \tag{2.4.15}$$

陀螺仪测得的角速度经过积分可以计算出角度，即

$$\theta_t=\int_t\omega\,\mathrm{d}t \tag{2.4.16}$$

互补滤波融合的公式为

$$G\cdot(\theta+\theta_t)+A\cdot\mathrm{A_{CC}} \tag{2.4.17}$$

更新姿态角，则融合后的角度为

$$\theta = G \cdot \left(\theta + \int_t \omega \, dt\right) + A \cdot \arccos\left(\frac{a_{cc}}{g}\right) \tag{2.4.18}$$

最终修正值 θ 即为融合后的角度。G 和 A 是根据时间常数计算出来的参数，且在应用中主要是调节这两个值的大小。在短时间内以陀螺仪为主，长时间上是趋近于加速度计的加权平均值。互补滤波器的流程框图如图 2.4.4 所示。

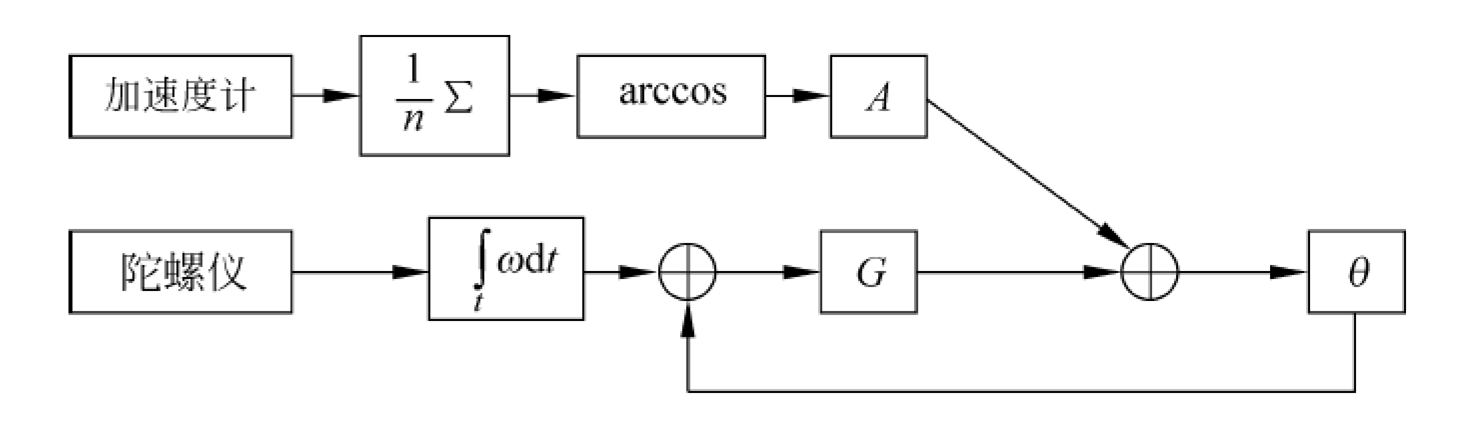


图 2.4.4 互补滤波器流程框图

常用于加速度计和陀螺仪数据融合的算法有一阶互补滤波算法、二阶互补滤波算法和卡尔曼滤波算法。下面对各种算法进行说明。

1. 一阶互补滤波

一阶互补滤波指的是对加速度计和陀螺仪计算而得的角度数据进行简单加权。实现一阶互补滤波的程序代码如下：

```
K = 0.075;                                        //对加速度计取值的权重
float A = K / (K + dt);
Com_angle = A * (Com_angle + omega * dt) + (1 - A) * angleA;
//Com_angle 为融合角度值,omega 为角速度,dt 为采样间隔,angleA 为加速度计求得角度
```

2. 二阶互补滤波

二阶互补滤波则是将加速度信号作为参考值，将融合计算得到的角度与加速度计算出的角度的偏差量加入积分中。代码实现如下：

```
K = 0.5;
float x1 = (angleA - Com2_angle) * K * K;
y1 = y1 + x1 * dt;
float x2 = y1 + 2 * K * (angleA - Com2_angle) + omega;
Com2_angle = Com2_angle + x2 * dt;
```

第 3 章



四旋翼飞行器常用通信协议

3.1 常用通信接口协议

3.1.1 SPI

1. SPI 总线接口概述

串行外设接口(Serial Peripheral Interface,SPI)总线系统是一种同步串行外设接口,是 Motorola 公司首先在其 MC68HCXX 系列处理器上定义的。SPI 接口主要应用于 EEPROM、Flash、实时时钟、AD 转换器,以及数字信号处理器和数字信号解码器之间,允许 MCU 与各种外围设备以串行方式进行通信、数据交换[9]。SPI 系统可直接与各个厂家生产的多种标准外围器件直接相连,一般使用 4 条线。

① 串行时钟线 SCLK。其主要作用是主设备往从设备传输时钟信号,控制数据交换的时机以及速率。

② 主机输入/从机输出数据线 MISO。也称为 Rx 通道,作为数据的入口,主要用于 SPI 设备接收数据。

③ 主机输出/从机输入数据线 MOSI。也称为 Tx 通道,作为数据的出口,主要用于 SPI 设备发送数据。

④ 从机使能信号 nSS(有的 IC 会标注为 CS,意为片选信号)。用于主设备片选从设备,使被选中的从设备能够被主设备所访问。

SPI 接口芯片带有中断信号线 INT 或 INT *,有的 SPI 接口芯片没有主机输出/从机输入数据线 MOSI。所以,SPI 系统总线只需 3～5 条数据线和控制线即可同具有 SPI

的各种 I/O 器件连接。

SPI 总线可在软件的控制下构成各种简单的或复杂的系统。例如，一个主 MCU 和几个从 MCU；几个从 MCU 相互连接构成多主机系统(分布式系统)；一个主 MCU 和一个或几个从 I/O 设备。大多数情况下，使用一个 MCU 作为主机，控制数据向一个或多个从外围器件的传送，从器件只能在主机的 SCLK 信号有时钟跳变时才能传送和采集信号。

图 3.1.1 是对 SPI 设备间通信的一个简单的描述，其中的组件及其含义如下。

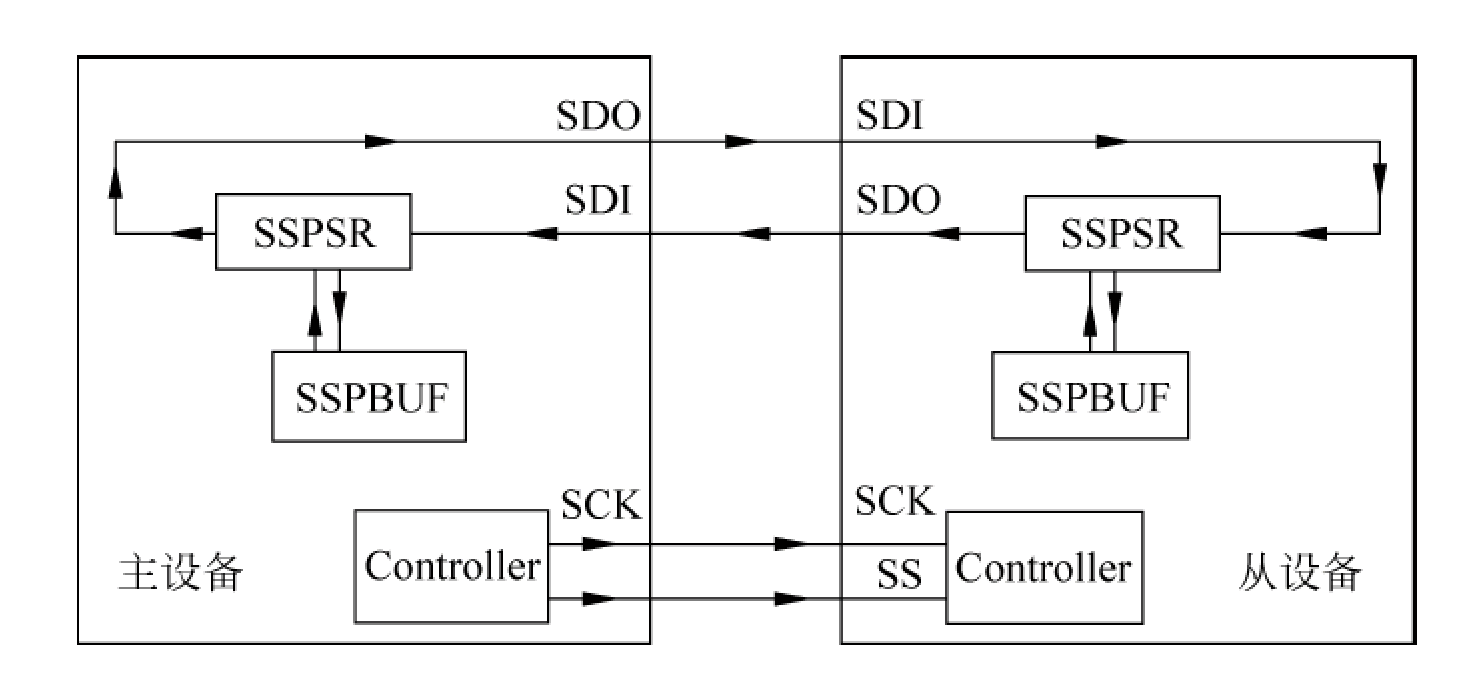


图 3.1.1 SPI 设备通信结构

① SSPBUF(Synchronous Serial Port Buffer)，泛指 SPI 设备里面的内部缓冲区，一般在物理上是以 FIFO 的形式保存传输过程中的临时数据。

② SSPSR(Synchronous Serial Port Register)，泛指 SPI 设备里面的移位寄存器(Shift Regitser)，它的作用是根据设置好的数据位宽(bit-width)把数据移入或者移出 SSPBUF。

③ Controller，泛指 SPI 设备里面的控制寄存器，可以通过配置它们来设置 SPI 总线的传输模式。

SPI 典型结构组成如图 3.1.2 所示。

2. SPI 总线接口数据传送

SPI 是一个环形总线结构，其时序简单，主要是在 SCLK 的控制下，两个双向移位寄存器进行数据交换，其时序结构如图 3.1.3 所示。上升沿发送数据，下降沿接收数据。其数据的传输格式是高位(MSB)在前，低位(LSB)在后。上升沿到来的时候，SDO 上最高位的电平将被发送到从设备的寄存器中；下降沿到来的时候，SDI 上的电平将被接收到主设备的寄存器中。一个完整的传送周期是 16 位，即两个字节，因为，首先主机要发

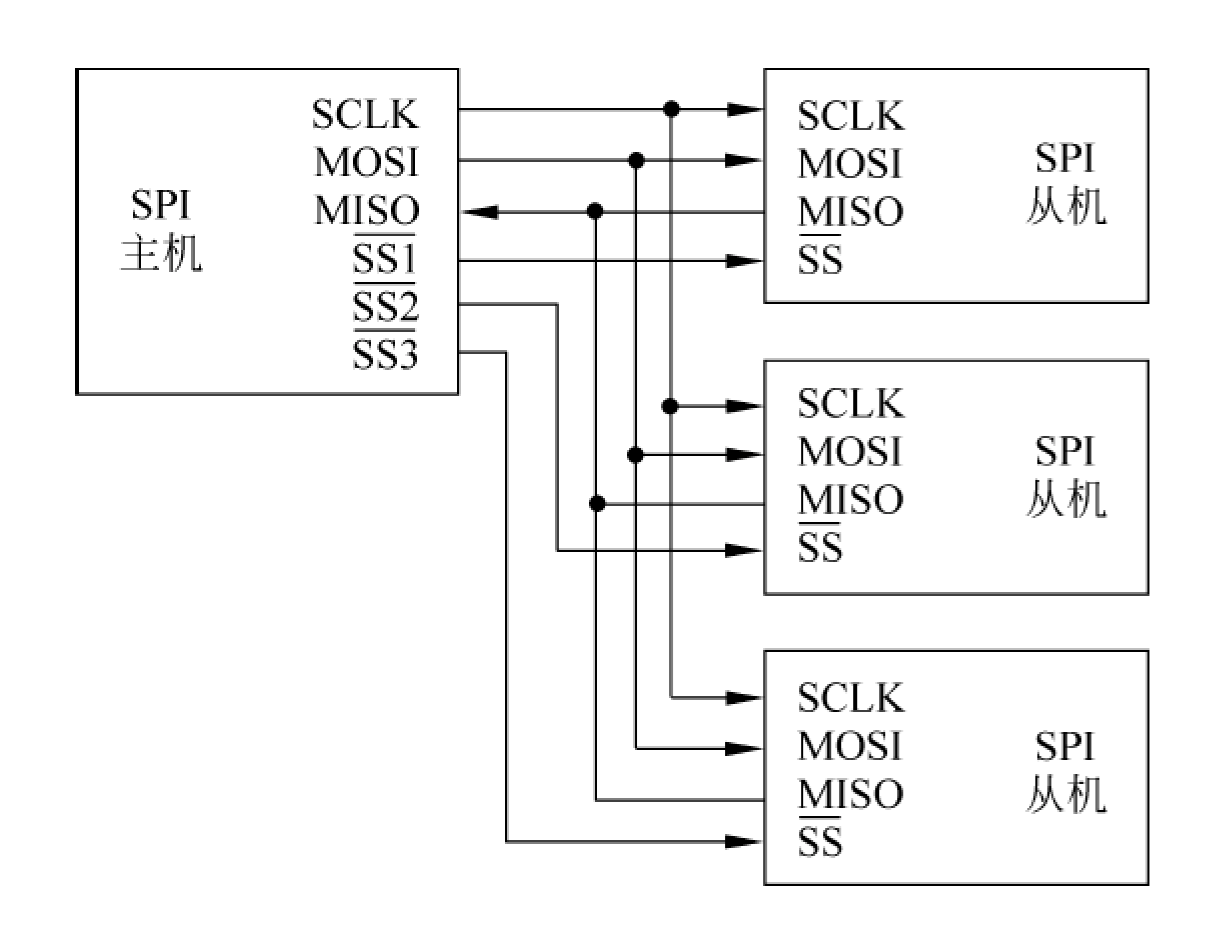


图 3.1.2　SPI 总线结构(1 主机+3 独立从机)

送命令过去,然后从机根据主机的命令准备数据,主机在下一个 8 位时钟周期才把数据读回来。

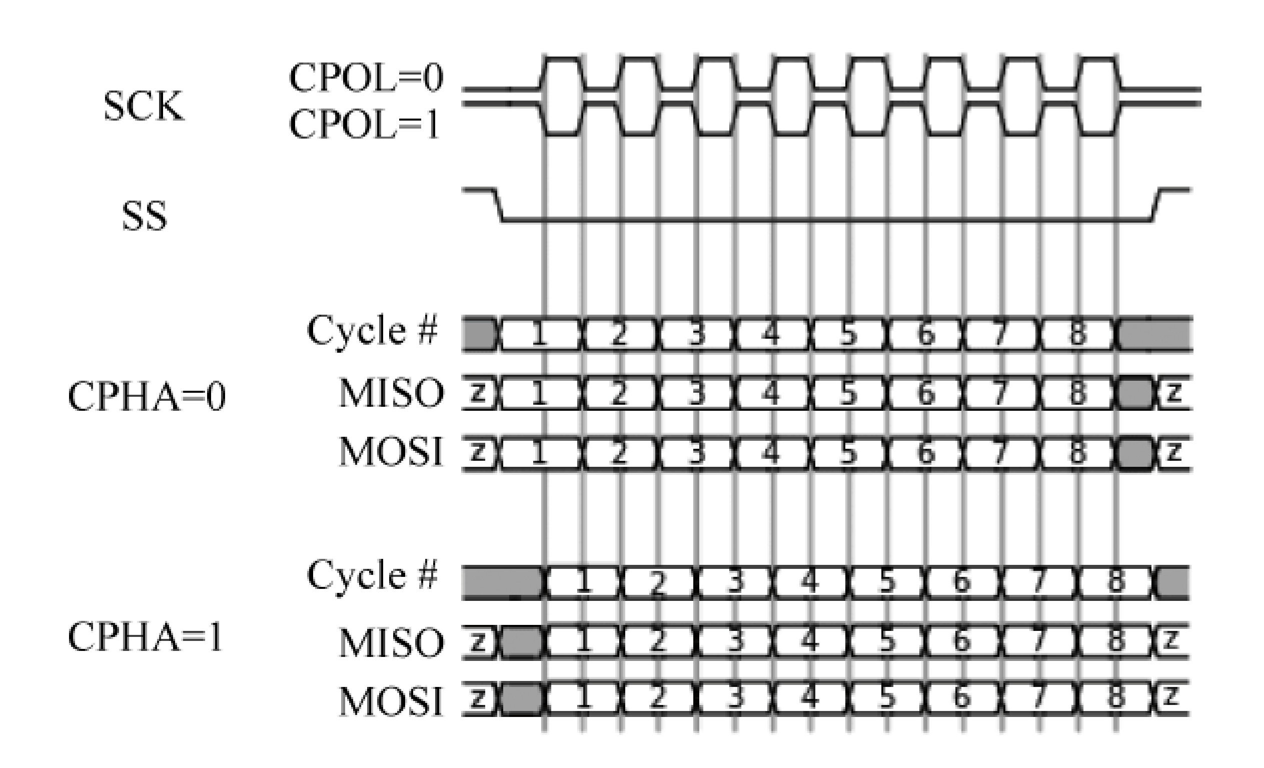


图 3.1.3　SPI 总线时序

SPI 模块为了和外设进行数据交换,根据外设工作要求,其输出串行同步时钟极性和相位可以进行配置,时钟极性(CPOL)对传输协议没有大的影响。如果 CPOL=0,串行同步时钟的空闲状态为低电平;如果 CPOL=1,串行同步时钟的空闲状态为高电平。时钟相位(CPHA)能够配置用于选择两种不同的传输协议之一进行数据传输。如果 CPHA=0,在串行同步时钟的第一个跳变沿(上升或下降)数据被采样;如果 CPHA=1,在串行同步时钟的第二个跳变沿(上升或下降)数据被采样。SPI 主模块和与之通信的外设时钟相位和极性应该一致。

因此根据 CPOL 和 CPHA 的设置，SPI 共可分为 4 种模式，如表 3.1.1 所示。

表 3.1.1　SPI 4 种模式

SPI 模式	时钟极性(CPOL)	时钟相位(CPHA)
0	0	0
1	0	1
2	1	0
3	1	1

另一种常用的模式表示方式为如“(0,1)”即表明 CPOL=0，CPHA=1。下面给出一段用软件实现 SPI 串行通信的例程。

```
int main(void)
{
SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_10MHZ);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI1);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlGPIOAHBEnable(GPIO_PORTF_BASE);           //挂在 AHB 高性能总线上

    GPIOPinConfigure(GPIO_PF2_SSI1CLK);
    GPIOPinConfigure(GPIO_PF3_SSI1FSS);
    GPIOPinConfigure(GPIO_PF1_SSI1TX);
    GPIOPinTypeSSI(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_3|GPIO_PIN_2);
    //SSI1_BASE/时钟频率/时钟协议,moto 代表 SPI 协议/工作模式: 主机模式/位速率 4M/数据
帧位数 16
    SSIConfigSetExpClk(SSI1_BASE, SysCtlClockGet(), SSI_FRF_MOTO_MODE_1, SSI_MODE_
MASTER, 16000000, 16);
    SSIEnable(SSI1_BASE);
    while(1){
    SSIDataPut(SSI1_BASE, 0x2468);
    }
}
```

以上程序能够实现通过 SPI 接口发送消息，在使用 SPI 发送消息之前，首先要对 SPI 功能进行配置，这也是以上程序的重点所在，SPI 初始化程序设计流程框图如图 3.1.4 所示。

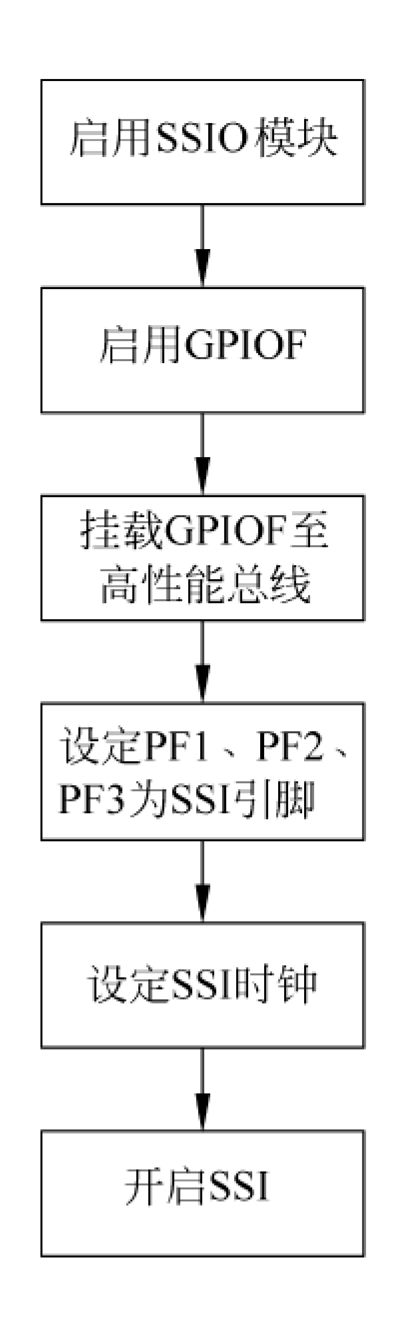


图 3.1.4　Tm4C 芯片 SPI 配置流程框图

3.1.2　I²C

1. I²C 总线接口概述

I²C(Inter-Integrated Circuit)是飞利浦(Philips)公司推出的一种高性能芯片间串行传输总线,与 SPI 不同,它仅以两根线实现了完善的全双工同步数据传送,可以极方便地构成多机系统和外围器件扩展系统[10]。其总线的基本结构如图 3.1.5 所示,I²C 总线采用了器件地址的设置方法,通过软件寻址完全避免了器件的片选寻址的弊端,从而使硬件系统具有更简单、更灵活的扩展方法。

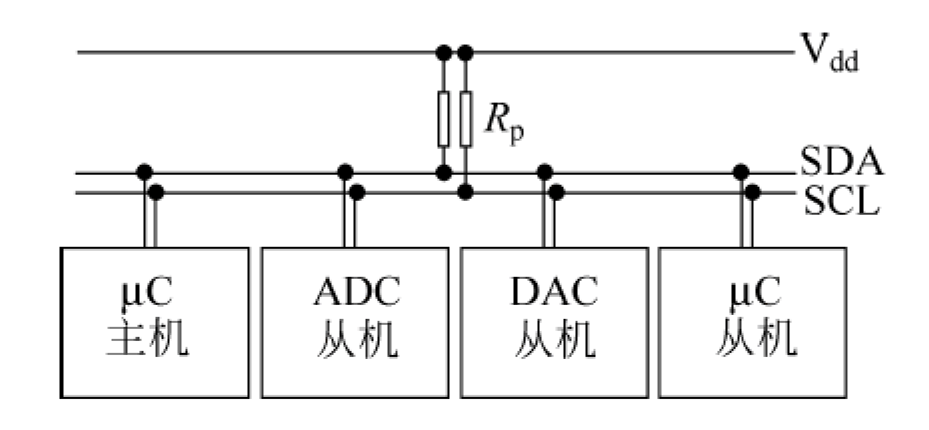


图 3.1.5　I²C 总线的基本结构

I^2C 总线进行数据传输时只需两根信号线，一根是双向的数据线 SDA，另一根是时钟线 SCL。所有连接到 I^2C 总线上的设备，其串行数据都连接到总线的 SDA 线上，而各设备的时钟线均接到总线的 SCL 线上。

I^2C 总线是一个多主机总线，即一个 I^2C 总线可以有一个或多个主机，总线运行由主机控制。这里所说的主机是指启动数据的传送（发起始信号）、发出时钟信号、传送结束时发出终止信号的设备。通常，主机由各类单片机或其他微处理器担当。被主机寻访的设备为从机，它可以是各种单片机或其他微处理器，也可以是其他器件，如存储器、LED 或 LCD 驱动器、A/D 或 D/A 转换器、时钟日历器件等。

I^2C 总线为双向同步串行总线，因此 I^2C 总线接口内部为双向传输电路。总线接口输出为开漏结构，所以总线上必须有上拉电阻。

为了进行通信，每个连接到 I^2C 总线上的设备都有一个唯一的地址。主机与从机之间的数据传送可以是由主机发送到总线上的其他设备，这时主机称为发送器。从总线上接收数据的设备为接收器。

在多主机系统中，可能同时有几个主机企图启动总线传送数据。为了避免混乱，保证数据的可靠传送，任意时刻总线只能由某一台主机控制，所以，I^2C 总线要通过总线裁决，来决定由哪一台主机控制总线。若有两个或两个以上的主机企图占用总线，一旦一个主机送“1”，而另一个或多个送“0”，送“1”的主机退出竞争。在竞争过程中，时钟信号是各个主机产生异步时钟信号线“与”的结果。

I^2C 总线上的时钟总是对应于主机的。传送数据时，每个主机产生自己的时钟，主机产生的时钟仅在慢速的从机拉宽低电平加以改变或在竞争中失败而改变。

当总线空闲时，两根总线均为高电平。连接到总线上的器件其输出级必须是漏极或集电极开路，任意设备输出的低电平都将使总线的信号变低，即各设备的 SDA 及 SCL 都是线与的关系。

2. I^2C 总线的数据传送

1）总线上数据的有效性

在 I^2C 总线上，每一位数据位的传送都与时钟脉冲相对应，逻辑“0”和逻辑“1”的信号电平取决于相应的正端电源 V_{DD} 的电压。I^2C 总线进行数据传送时，在时钟信号为高电平期间，数据线上必须保持有稳定的逻辑电平状态，高电平为数据 1，低电平为数据 0。只有在时钟线低电平期间，才允许数据线上的电平状态变化，如图 3.1.6 所示。



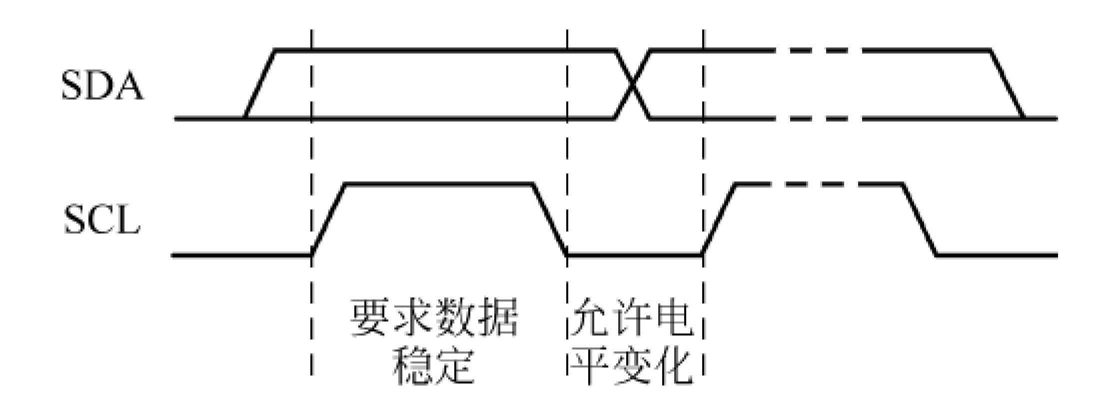


图 3.1.6　数据位的有效性规定

2）数据传送的起始信号和终止信号

起始信号和终止信号如图 3.1.7 所示。

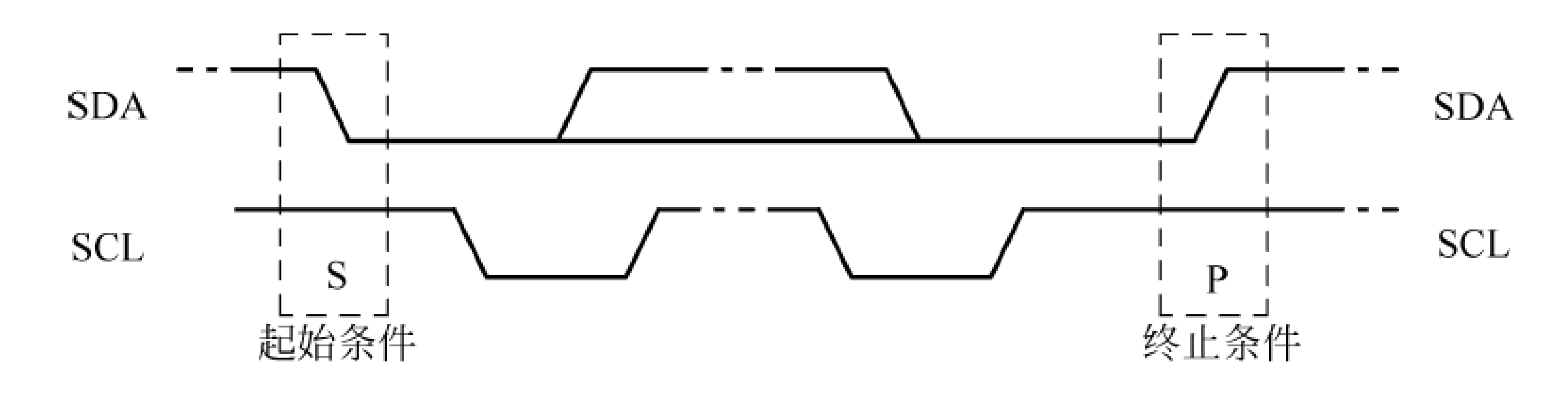


图 3.1.7　起始信号和终止信号

根据 I^2C 总线协议的规定：SCL 线为高电平期间，SDA 由高电平向低电平的变化表示起始信号；SCL 线为高电平期间，SDA 线由低电平向高电平的变化表示终止信号。

起始信号和终止信号都是由主机发出的，在起始信号产生后，总线就处于被占用的状态；在终止信号产生一定时间后，总线就处于空闲状态。连接到 I^2C 总线上的设备若具有 I^2C 硬件接口，则很容易检测到起始信号和终止信号。对于不具备 I^2C 总线接口的 MCU 来说，为了能准确检测起始信号和终止信号，必须保证在总线的一个时钟周期内对数据线至少采样两次。

从机收到一个完整的数据字节后，有可能需要完成一些其他工作，如处理内部中断服务等，可能使它无法立刻接收下一个字节。这时从机可以将 SCL 线拉成低电平，从而使主机处于等待状态，直到从机准备好可以接收下一个字节时，再释放 SCL 线使之为高电平，数据传送继续进行。

3）数据传送格式

（1）字节传送与应答

利用 I^2C 总线进行数据传送时，对传送的字节数是没有限制的，但是每一个字节必须保证是 8 位长度，并且首先发送的数据位为最高位，每传送一个字节数据后都必须跟随

一位应答信号，与应答信号相对应的时钟由主机产生，主机必须在这一时钟位上释放数据线，使其处于高电平状态，以便从机在这一位上送出应答信号，如图 3.1.8 所示。

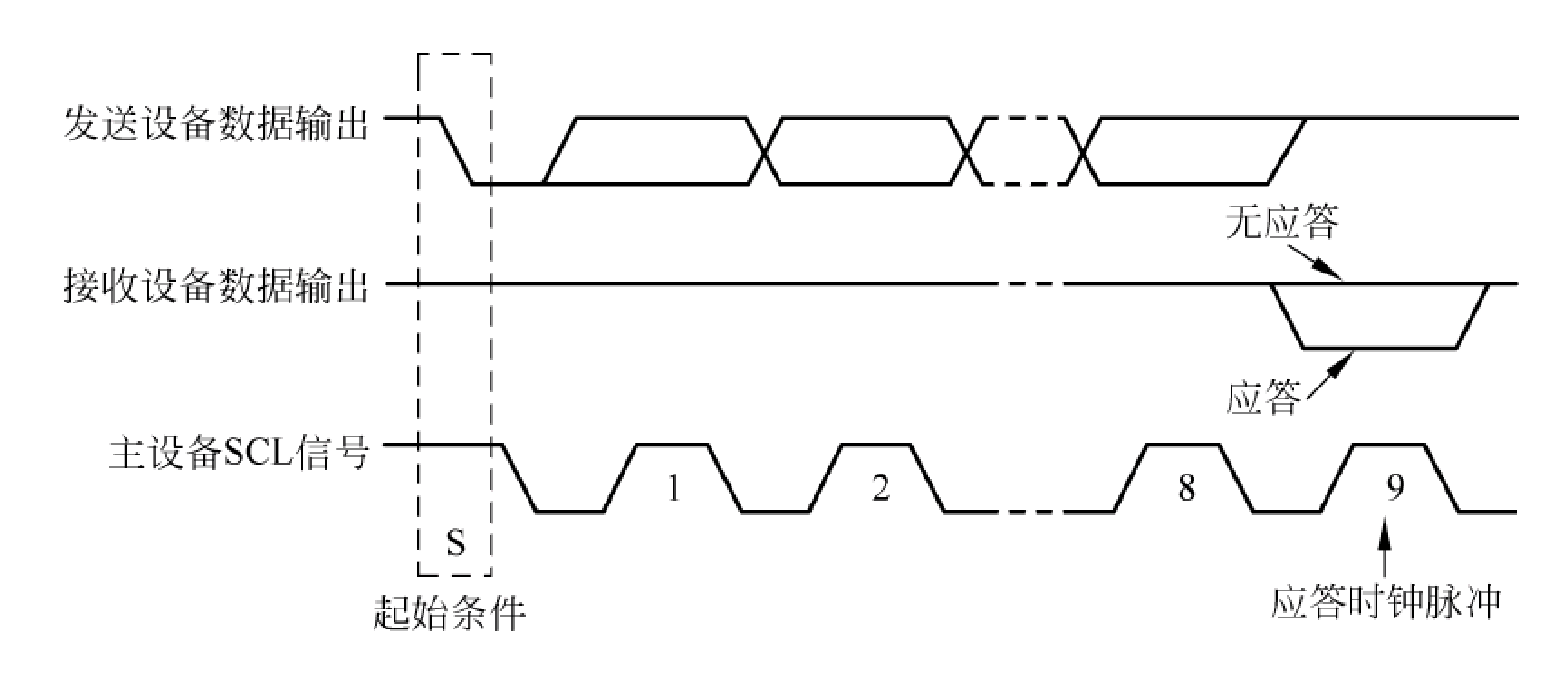


图 3.1.8 I^2C 总线应答时序

应答信号在第 9 个时钟位上出现，从机输出低电平为应答信号，表示继续接收，若从机输出高电平则为非应答信号，表示结束接收。由于某种原因，从机不对主机寻址信号应答时（如从机正在进行实时性的处理工作而无法处理接收总线上的数据），它必须释放总线，将数据线置于高电平，然后由主机产生一个终止信号以结束总线的数据传送。

如果从机对主机进行了应答，但在数据传送一段时间后无法继续接收更多的数据时，从机可以通过发送非应答信号通知主机，主机则应发出终止信号来结束数据的继续传送。

当主机接收数据时，它收到最后一个数据字节后，必须向从机发送一个非应答信号，使从机释放 SDA 线，以便主机产生终止信号，从而停止数据传送。

（2）数据传送格式

I^2C 总线上传输的数据信号是广义的，既包括地址信号，又包括真正的数据信号。I^2C 总线数据传送时必须遵守规定的数据传送格式，如图 3.1.9 所示。

按照总线规约，起始信号表明一次数据传送的开始，其后为寻址字节，寻址字节由高 7 位地址和最低 1 位方向位组成，高 7 位地址是被寻址的从机地址，方向位是表示主机与从机之间的数据传送方向，方向位为“0”时表示主机发送数据“写”，方向位为“1”时表示主机接收数据“读”。在寻址字节后是将要传送的数据字节与应答位，在数据传送完后主机必须发送终止信号。但是，如果主机希望继续占用总线进行新的数据传送，则可以不产生终止信号，马上再次发出起始信号对另一从机进行寻址。

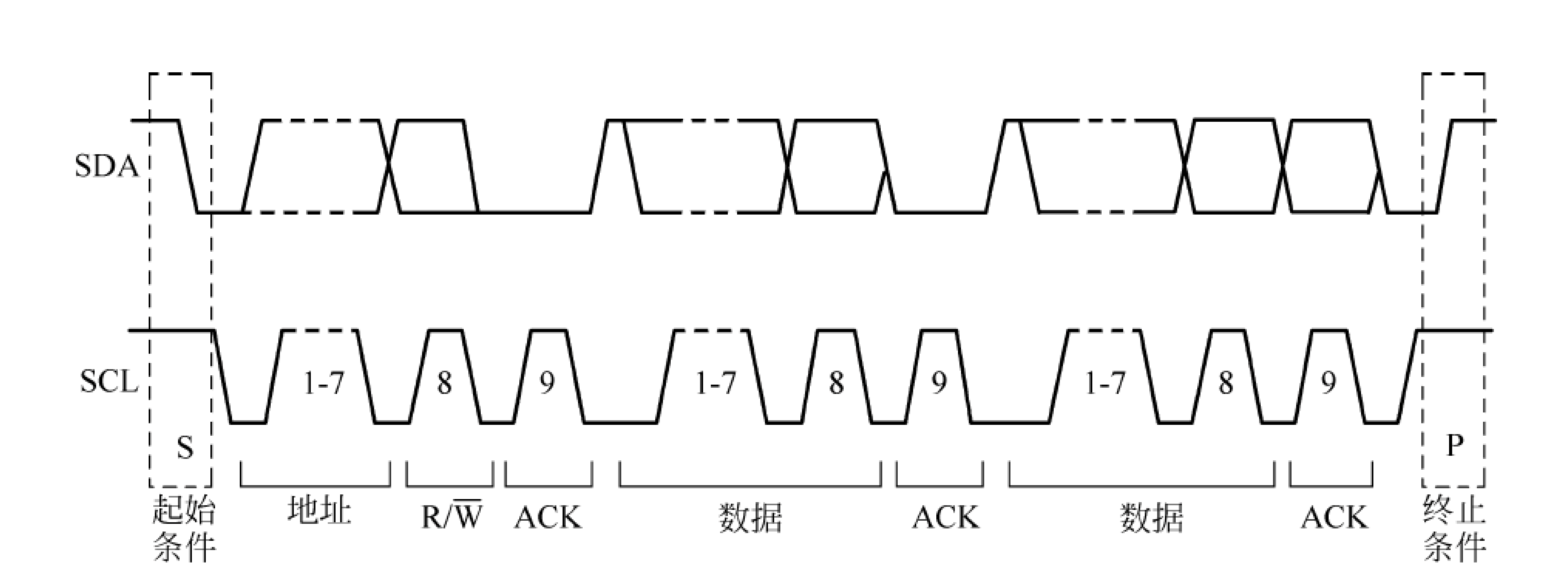


图 3.1.9　I^2C 总线一次完整的数据传输格式

因此，在总线的一次数据传送过程中，可以有几种读、写组合方式。

① 主机向从机发送 n 个数据，数据传送方向在整个传送过程中不变，如图 3.1.10 所示。

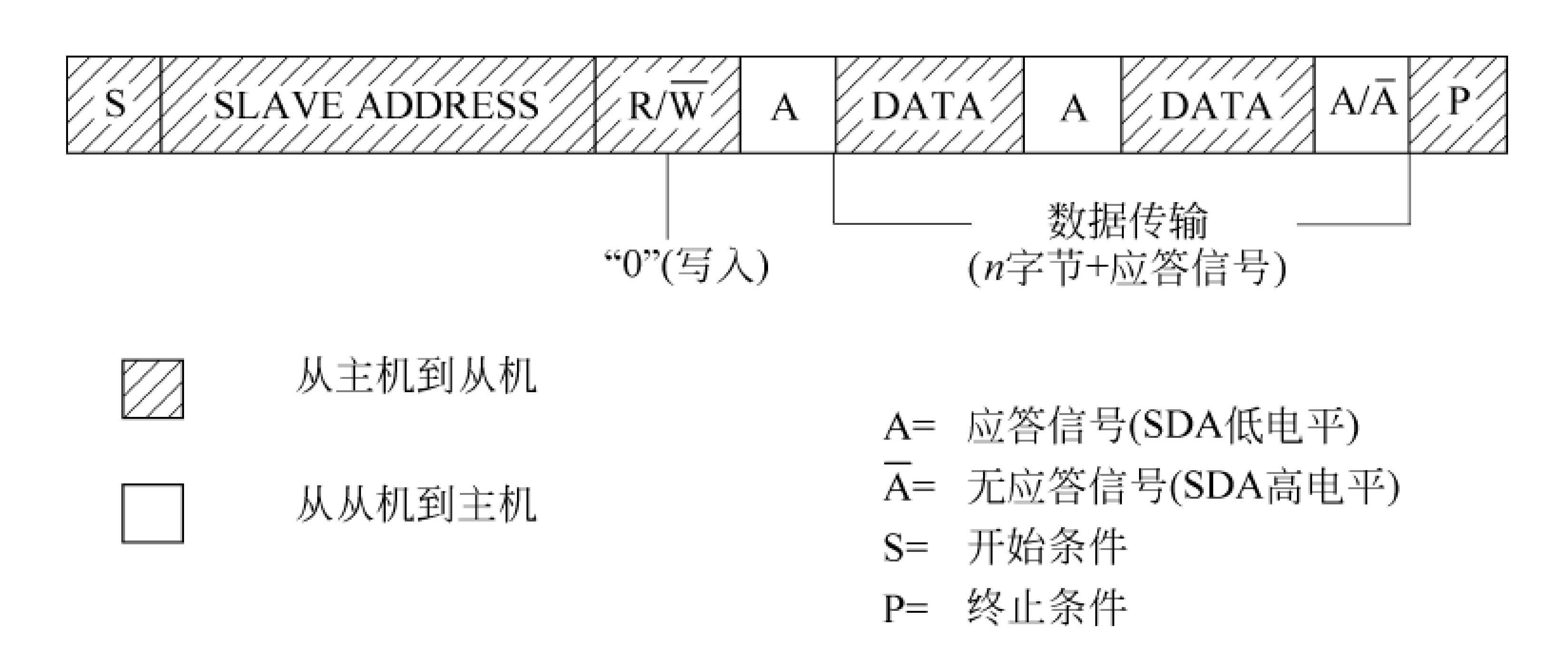


图 3.1.10　I^2C 数据传输方式 1

主机在第一个字节后，立即由从机读数据，在整个过程中，除寻址字节外，都是从机发送、主机接收，如图 3.1.11 所示。

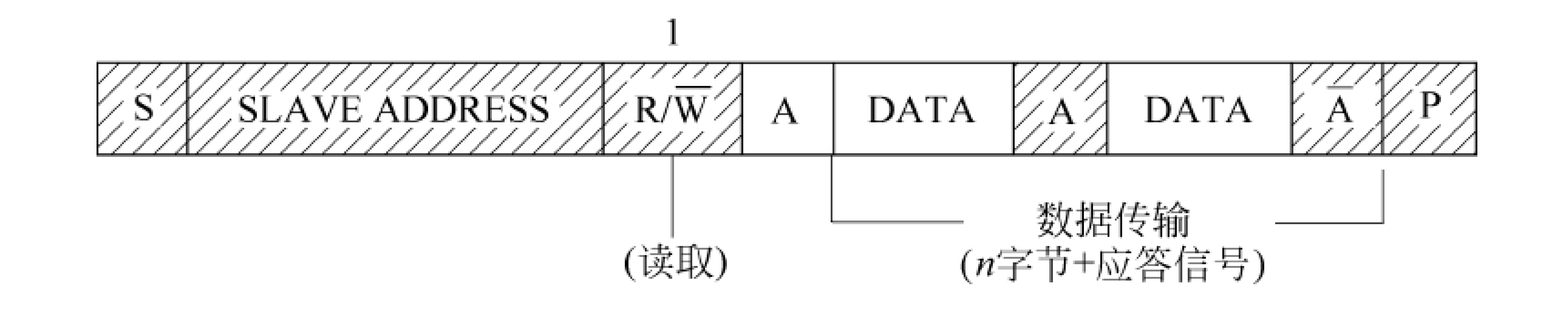


图 3.1.11　I^2C 数据传输方式 2

② 在传送过程中，当需要改变传送方向时，起始信号和从机地址都被重复产生一次，但两次读/写方向位正好反相，如图 3.1.12 所示。

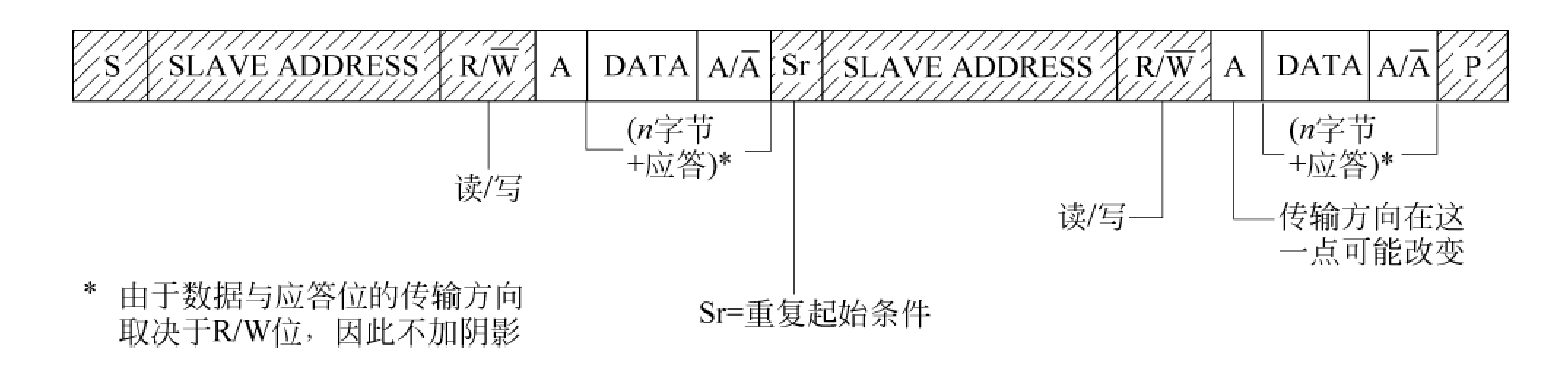


图 3.1.12 I²C 数据传输方式 3

由以上格式可见，无论采用哪种方式，起始信号、终止信号和地址均由主机发送，数据字节的传送方向由寻址字节中方向位规定；每个字节的传送都必须有应答信号位相随。

4）I²C 总线的寻址约定

I²C 总线是多主总线，总线上的各个主机都可以争用总线，在竞争中获胜者马上占有总线控制权。I²C 总线协议对有权使用总线的主机寻址做出了明确的规定：采用 7 位的寻址字节，寻址字节是起始信号后的第一个字节。

（1）寻址字节的位定义

从高到低分别为 7 位数据，最低位 LSB 为数据传送方向位，如图 3.1.13 所示。主机发送地址时，总线上的每个从机都将这 7 位地址码与自己的期间地址进行比较，如果相同则认为自己正被主机寻址，根据读/写位将自己确定为接收器或发送器。

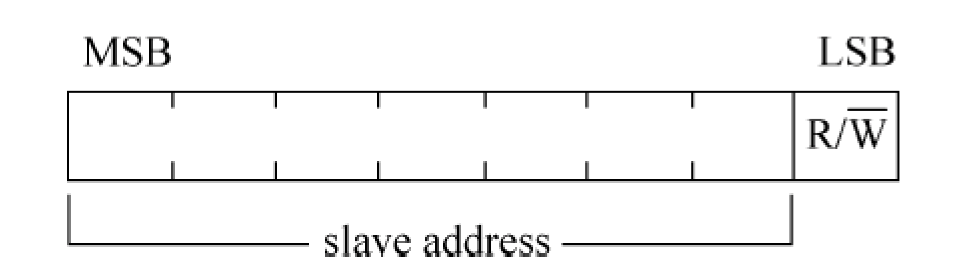


图 3.1.13 寻址字节

从机的地址是由一个固定部分和一个可编程部分组成。固定部分为器件的编号地址，表明了器件的类型，出厂时是固定的，不可更改；可编程部分为器件的呼叫地址，视硬件接线而定，引脚地址数决定了同一种器件可接入到 I²C 总线中的最大数目。如果从机为单片机，则 7 位地址为纯软件地址。

（2）寻址字节中的特殊地址

I²C 总线地址统一由 I²C 总线委员会实行分配，其中两组编号地址 0000 和 1111 已被保留，留作特殊用途，如表 3.1.2 所示。

表 3.1.2　I^{2}C 总线特殊地址

<table>
<tr><th>地址位</th><th>读/写位</th><th>用途</th></tr>
<tr><td>0000 000</td><td>0</td><td>通用广播地址</td></tr>
<tr><td>0000 000</td><td>1</td><td>起始地址</td></tr>
<tr><td>0000 001</td><td>×</td><td>CBUS 地址</td></tr>
<tr><td>0000 010</td><td>×</td><td rowspan="4">待定</td></tr>
<tr><td>0000 011</td><td>×</td></tr>
<tr><td>0000 1××</td><td>×</td></tr>
<tr><td>1111 1××</td><td>×</td></tr>
<tr><td>1111 0××</td><td>×</td><td>10 位从机地址</td></tr>
</table>

① 广播地址。起始信号之后第一个字节为“0000 0000”时称为通用广播地址。广播地址用于寻访接到 I^{2}C 总线上的所有器件,并向它们广播数据。

② 起始字节是提供给没有 I^{2}C 总线接口的单片机查询 I^{2}C 总线时使用的特殊字节。对于不具备 I^{2}C 总线接口的单片机,采用软件模拟时序的方法,也可以接入 I^{2}C 总线系统。

下面给出在本系统采用 Ti 飞行控制板芯片的开发板 TIVA C Launchpad 上实现 I^{2}C 主从模块消息互通的主模块程序设计流程图(如图 3.1.14 所示)。从模块程序设计与主模块类似,但不用设置时钟频率等。

3.1.3　USART

1. USART 功能概述

USART(Universal Synchronous/Asynchronous Receiver/Transmitter)是一个全双工通用同步/异步串行收发模块,该接口是一个高度灵活的串行通信设备。与 UART 相比,它增加了同步功能,能够提供主动时钟来实现同步通信。它提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。它支持同步单向通信和半双工单线通信。它还允许多处理器通信。

接口通过 3 个引脚与其他设备连接在一起,任何 USART 双向通信至少需要两个脚,即接收数据输入(RX)和发送数据输出(TX)。USART 通信具有以下特性。

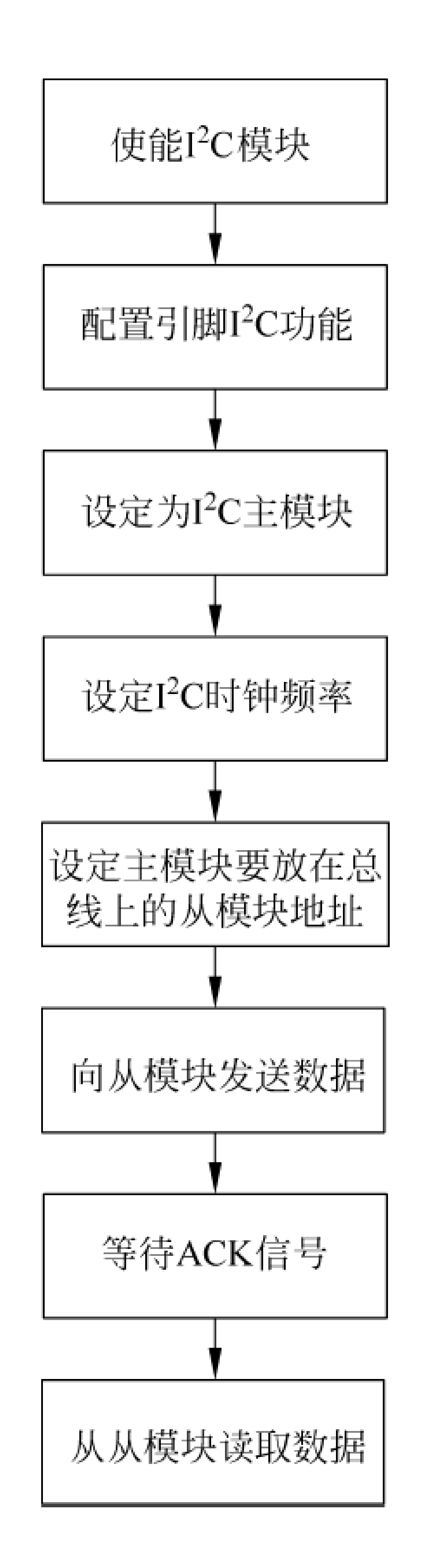


图 3.1.14　I²C 主模块程序设计流程框图

① 总线在发送或接收前应处于空闲状态。

② 一个起始位。

③ 一个数据字(8 位或 9 位),最低有效位在前。

④ 0.5、1.5 这两个的停止位,由此表明数据帧的结束。

⑤ 使用分数波特率发生器——12 位整数和 4 位小数的表示方法。

⑥ 一个状态寄存器(USART_SR)。

⑦ 一个数据寄存器(USART_DR)。

⑧ 一个波特率寄存器(USART_BRR),12 位整数和 4 位小数。

⑨ 一个智能卡模式下的保护时间寄存器(USART_GTPR)。

2. USART 数据传输

如图 3.1.15 所示,字长可以通过编程 USART_CR1 寄存器中的 M 位,选择成 8 位

或 9 位。在起始位期间，Tx 脚处于低电平，在停止位期间 Tx 脚处于高电平。空闲符号被视为完全由“1”组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位（“1”的位数也包括了停止位的位数）。

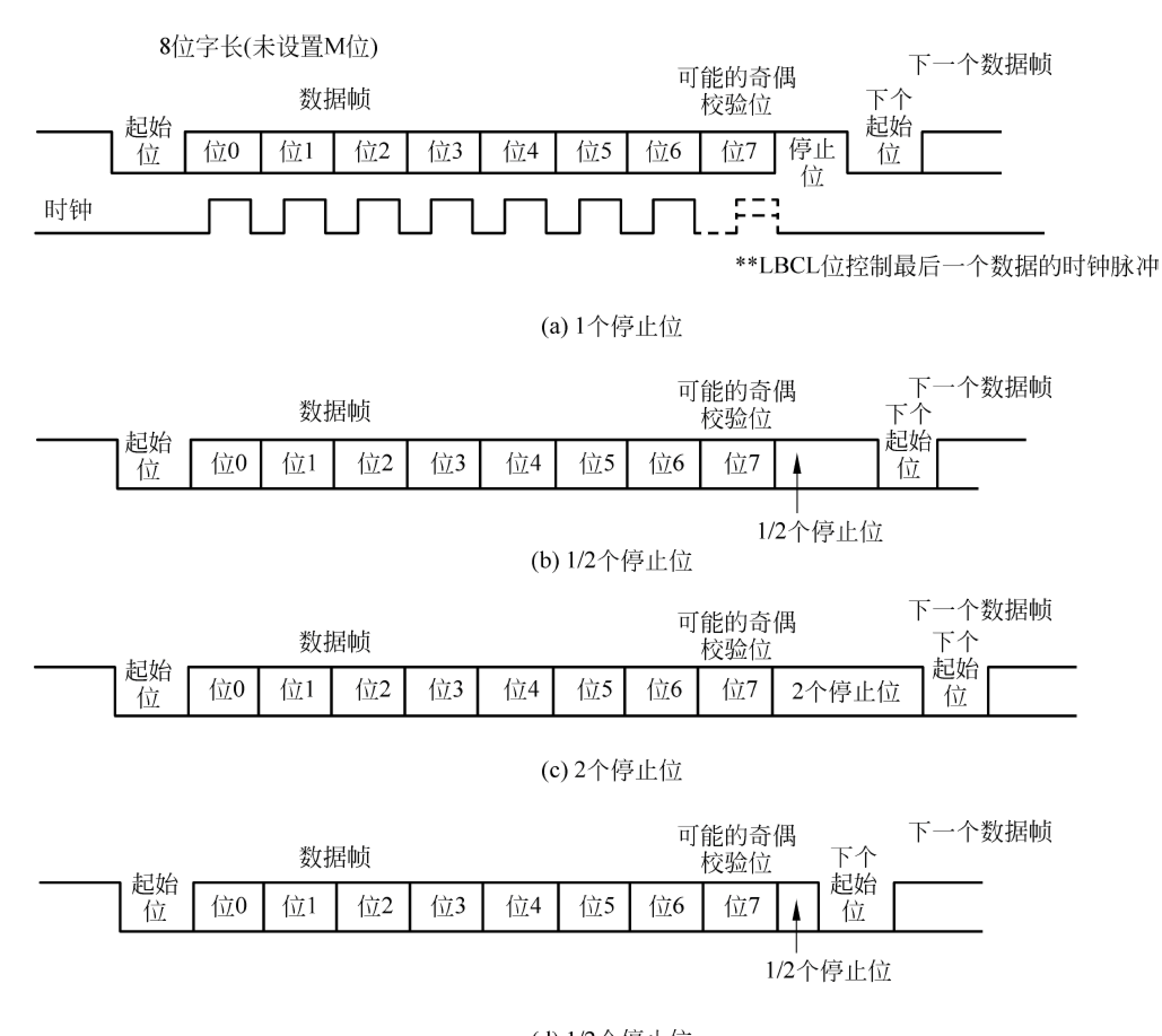


图 3.1.15　USART 的数据帧格式

断开符号被视为在一个帧周期内全部收到“0”（包括停止位期间，也是“0”）。在断开帧结束时，发送器再插入一或两个停止位（“1”）来应答起始位。发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位置位时，分别为其产生时钟。

在 TIVA C Launchpad 上进行 UART 串口通信的程序设计流程框图如图 3.1.16 所示。

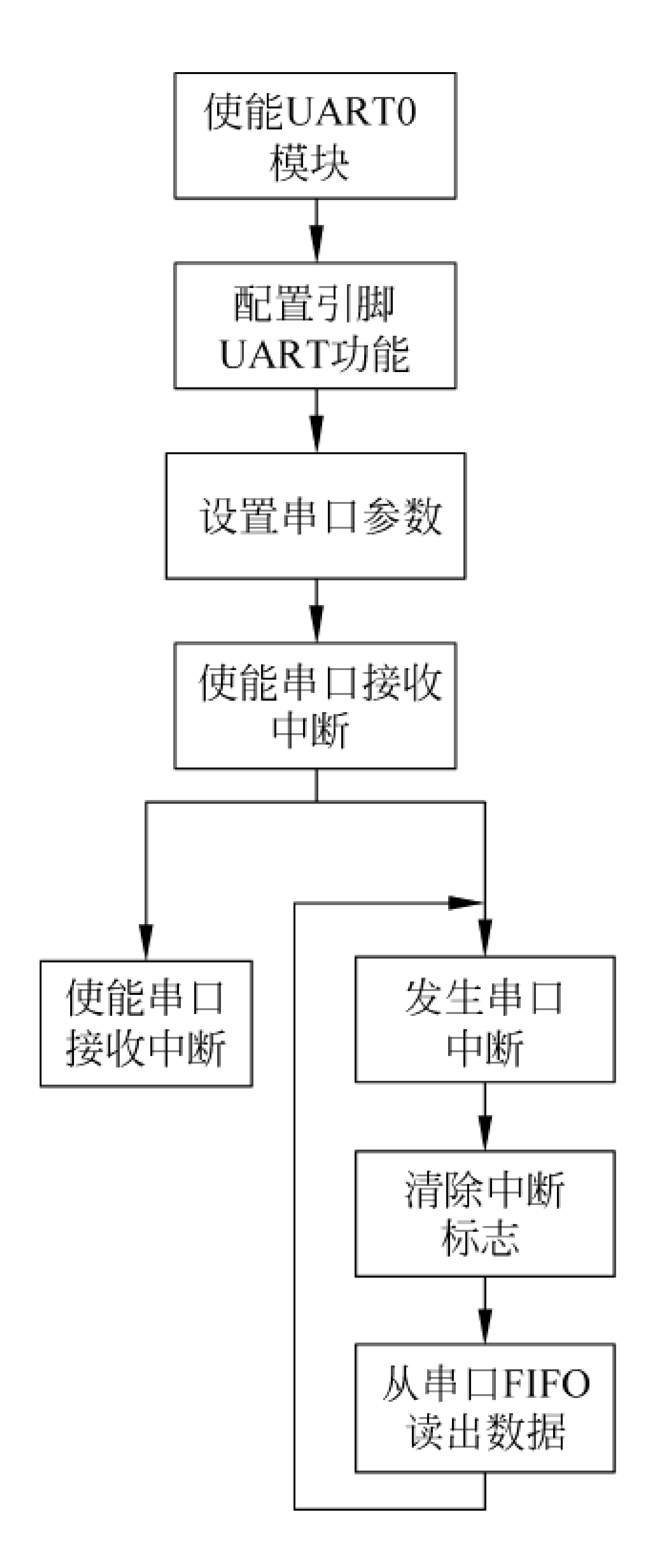


图 3.1.16 串口通信程序设计流程框图

3.2 常用 RC(Radio Controller)通信协议

3.2.1 PPM

PPM(Pulse Position Modulation,脉宽位置调制)调制信号控制脉冲序列中各脉冲的相对位置(即相位),使各脉冲的相对位置随调制信号变化。此时序列中脉冲幅度和宽度均不变。PPM 是遥控模型中比较通用的一种信号格式,是接收机收到遥控器发出的 4 通道信息后,输出给飞行控制模块的编码信号,飞行控制部分对其进行解码即可得到各个通道的控制量。

PPM 原理是通过检测给定频率的 PWM 信号的占空比来获取指令信号的。如图 3.2.1 所示,信号频率为 50Hz,一个周期为 20ms,每个信号脉宽 0～2ms; 1 帧 PPM

信号长度为 20ms，实际遥控最多可以容纳 9 个通道。一个通道的 2ms 时间内，脉宽仅在 1～2ms 内变化，即脉宽 1ms 表示停转，脉宽 2ms 表示满量程运作，其间各点按比例换算，如脉宽 1.5ms 表示 50％的量程。

PPM 信号波形如图 3.2.1 所示。

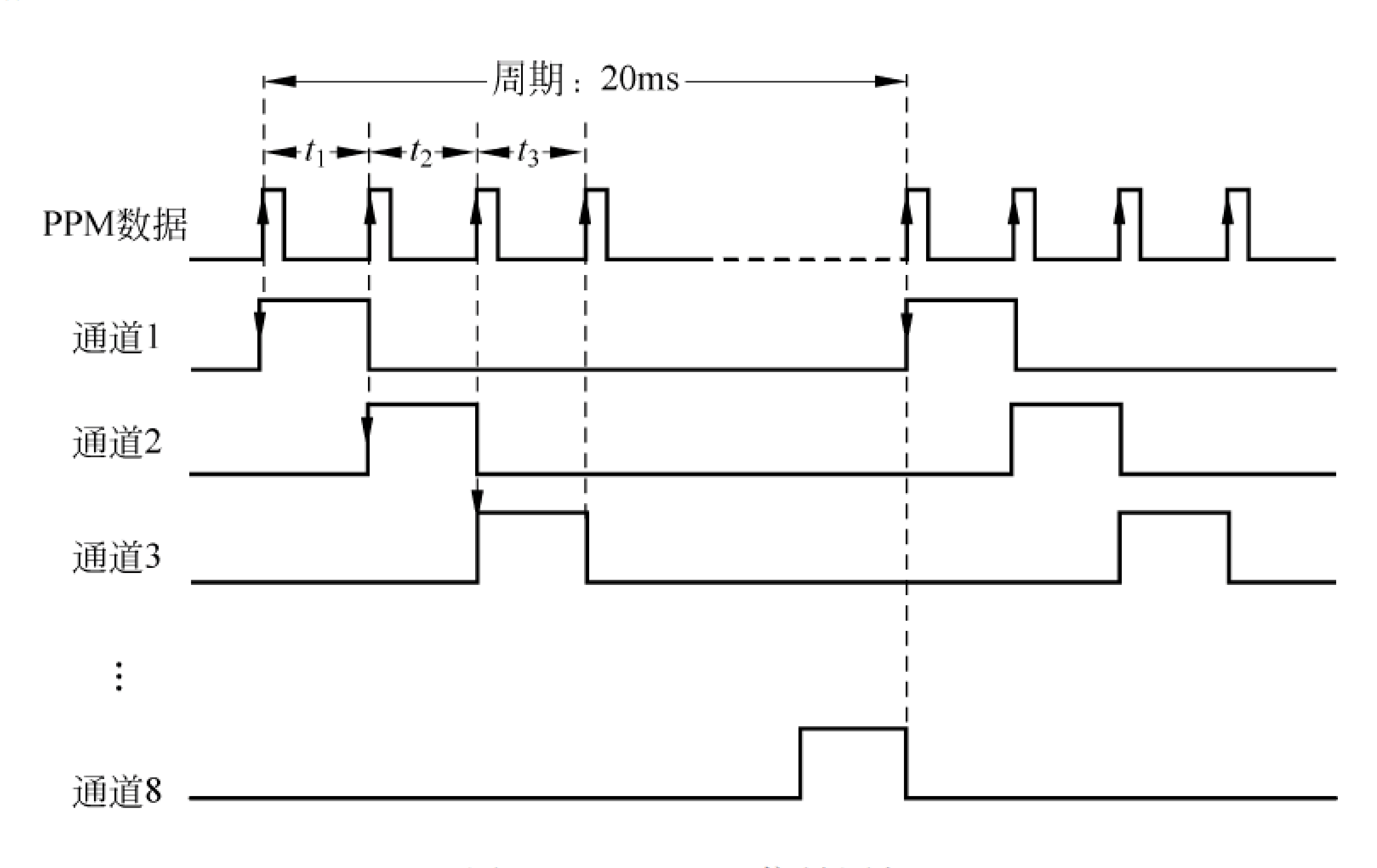


图 3.2.1　PPM 信号图解

3.2.2　PWM

PWM(Pulse Width Modulation，脉冲宽度调制)是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，一般变换后脉冲的周期固定，但脉冲的占空比会依模拟信号的大小而改变，如图 3.2.2 所示。PWM 被广泛应用在从测量、通信到功率控制与变换的许多领域中。

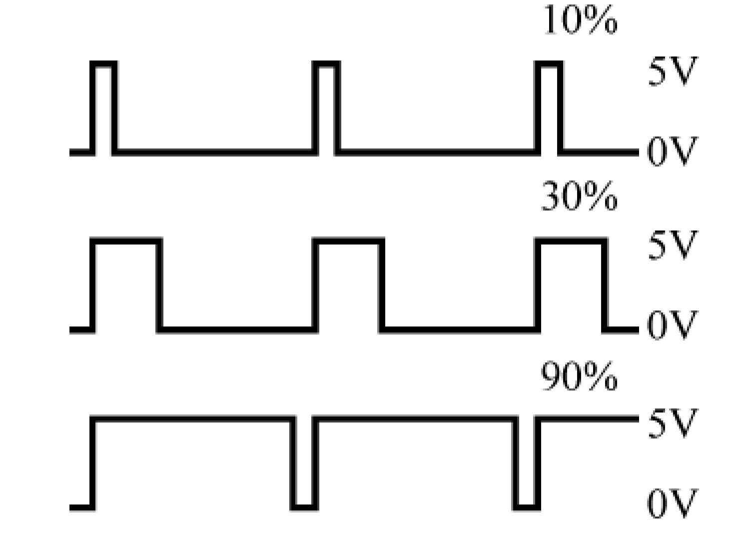


图 3.2.2　不同占空比的 PWM 波

在模拟电路中，模拟信号的值可以连续进行变化，在时间和值的幅度上几乎没有限制，基本上可以取任何实数值，输入与输出也呈线性变化。所以在模拟电路中，电压和电流可直接用来进行控制对象，如家用电器设备中的音量开关控制、采用卤素灯泡灯具的亮度控制等。但模拟电路有诸多的问题：例如，控制信号容易随时间漂移，难以调节；功耗大；易受噪声和环境干扰等。与模拟电路不同，数字电路是在预先确定的范围内取值，在任何时刻其输出只可能为 ON 和 OFF 两种状态，所以电压或电流会以通/断方式的重复脉冲序列加载到模拟负载。

PWM 技术是一种对模拟信号电平的数字编码方法，通过使用高分辨率计数器（调制频率）调制方波的占空比，从而实现对一个模拟信号的电平进行编码。其最大的优点是从处理器到被控对象之间的所有信号都是数字形式的，不需要再进行数模转换过程；而且对噪声的抗干扰能力也大大增强（噪声只有在强到足以将逻辑值改变时，才有可能对数字信号产生实质的影响），这也是 PWM 在通信等信号传输行业得到大量应用的主要原因。

模拟信号能否使用 PWM 进行编码调制，仅依赖带宽，这即意味着只要有足够的带宽，任何模拟信号值均可以采用 PWM 技术进行调制编码，一般而言，负载需要的调制频率要高于 10Hz，在实际应用中，频率在 1～200kHz 之间。

在信号接收端，需将信号解调还原为模拟信号，目前在很多微型处理器内部都包含有 PWM 控制器模块。

PWM 控制技术以其控制简单、灵活和动态响应好的优点而成为电力电子技术最广泛应用的控制方式，也是人们研究的热点。由于当今科学技术的发展已经没有了学科之间的界限，结合现代控制理论思想或实现无谐振波开关技术将会成为 PWM 控制技术发展的主要方向之一。其根据相应载荷的变化调制晶体管基极或 MOS 管栅极的偏置，来实现晶体管或 MOS 管导通时间的改变，从而实现开关稳压电源输出的改变。这种方式能使电源的输出电压在工作条件变化时保持恒定，是利用微处理器的数字信号对模拟电路进行控制的一种非常有效的技术。

对于具有 PWM 模块的芯片，可以方便地通过硬件方式输出 PWM 波。下面给出配置 PWM 功能的程序设计流程框图（如图 3.2.3 所示）。

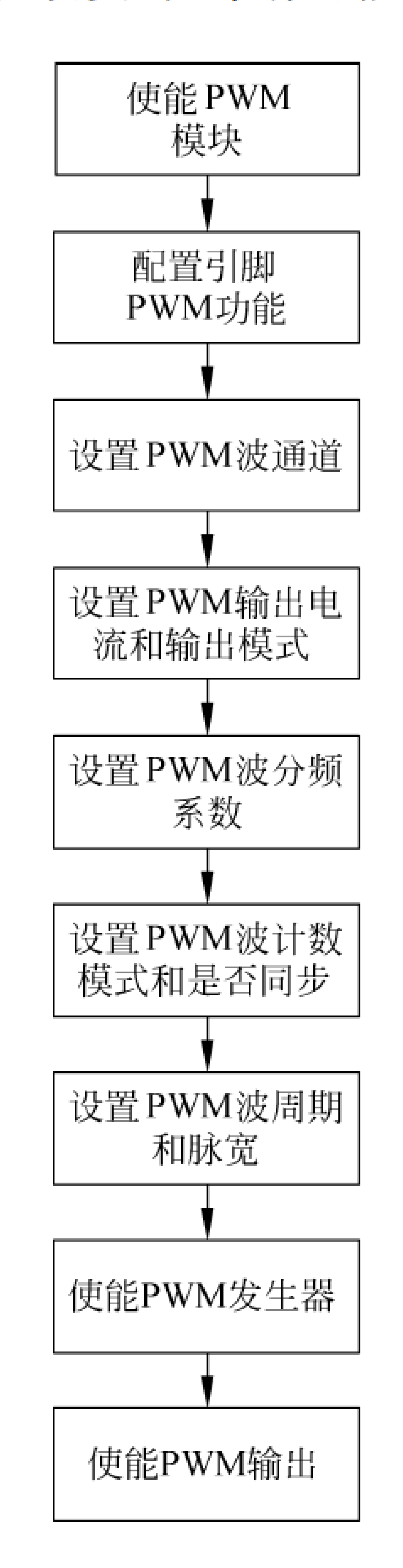
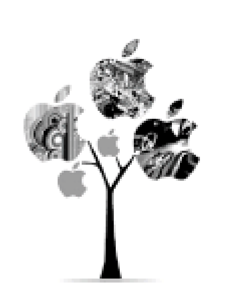


图 3.2.3　PWM 功能配置流程框图

第 4 章



四旋翼飞行器硬件系统设计

4.1 四旋翼飞行器系统硬件总体结构

这里研发四旋翼飞行器系统的硬件主要由地面控制板和飞行控制板两部分组成。PC 上位机和遥控器的控制信号经地面控制板处理后，再通过 nRF 无线模块与飞行控制板进行无线通信。

其硬件结构框图如图 4.1.1 所示。

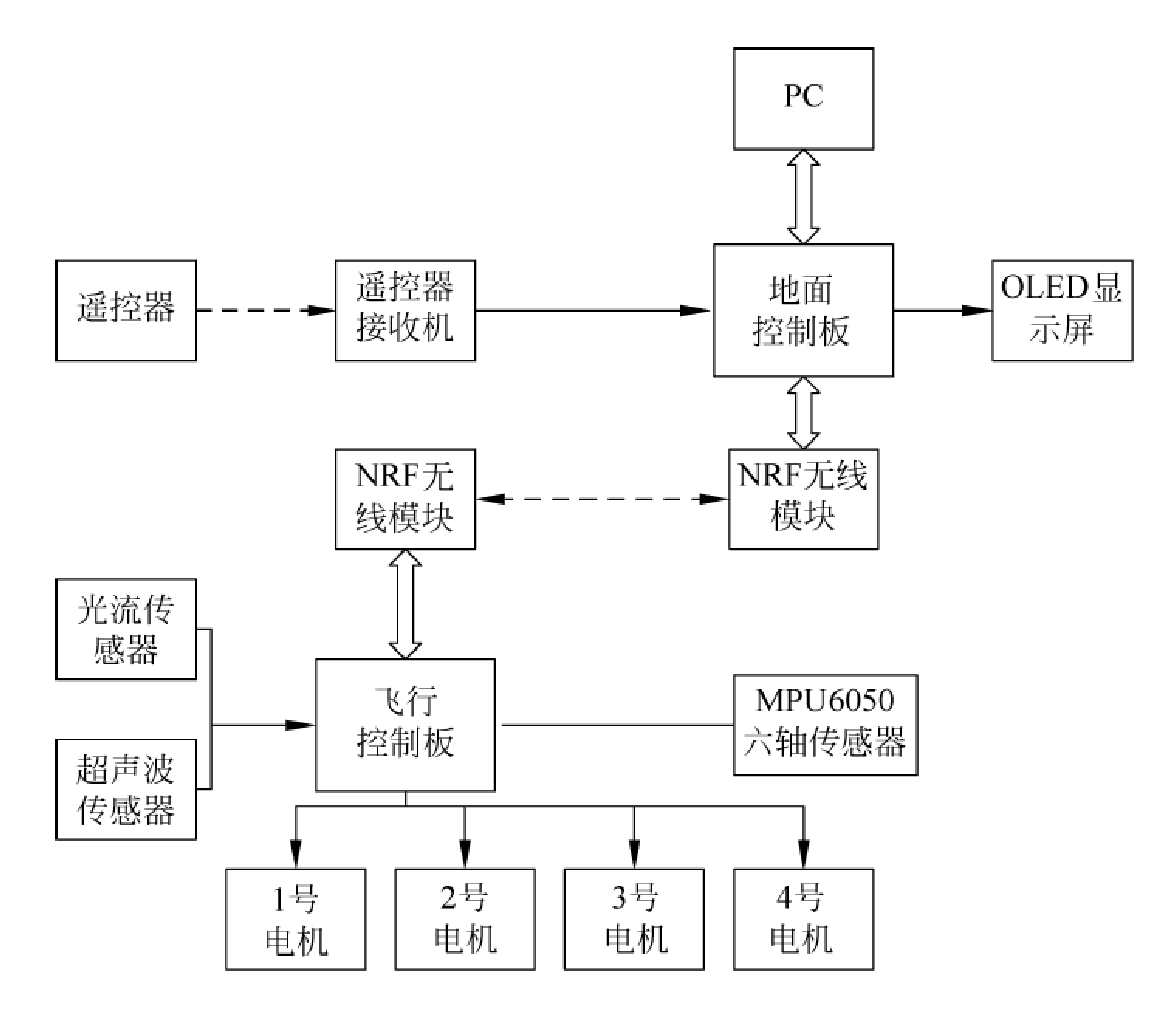


图 4.1.1　四旋翼飞行器系统硬件结构框图

四旋翼无人机实物如图 4.1.2 所示。



图 4.1.2　四旋翼飞行器实物

4.1.1　主控板

飞行控制板的主要功能有如下几个。

① 通过电机驱动程序调整飞行控制板输出四路 PWM 波的占空比,从而改变 4 个电机的转速来实现对四旋翼飞行器运动状态的控制。

② 飞行器传感器数据采集及处理,包括六轴传感器的姿态数据、超声波模块测量的高度数据、光流传感器的位移数据。

③ 与地面控制板的交互通信,接收地面控制站发送来的遥控器数据以及其他控制命令,同时向地面站反馈自身各项状态数据。

地面控制板(如图 4.1.3 所示)的主要功能如下。

① 实时采集遥控器数据,采集遥控器的 PPM 信号并解析成真正的遥控数据,实现遥控器对四旋翼飞行器的控制。

② 实现与飞行控制板之间的交互通信,包括传送遥控数据和控制命令数据,接收飞行器的状态数据,如超声波传送的高度数据。

③ 在 LED 屏幕上动态显示飞行器的飞行状态数据。

飞机与地面站之间能够通过 nRF 进行无线通信,上位机与地面站之间通过 USB 数据线建立有线通信。另外,还可以通过遥控器对飞机进行控制。

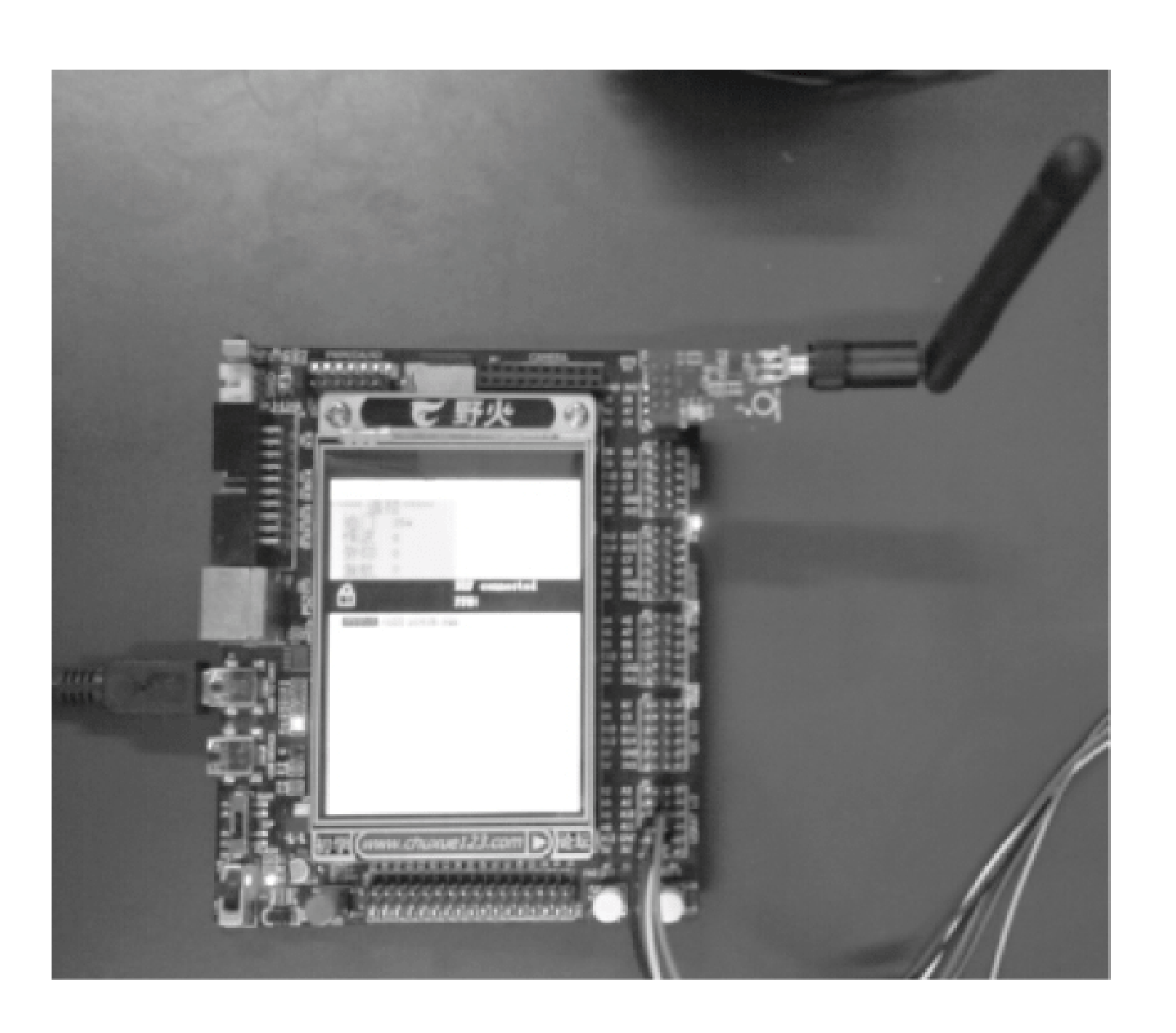


图 4.1.3　地面控制板实物

1. 微处理器模块

飞行控制板主芯片采用了美国得州仪器公司 Tiva 系列的 TM4C123G 微控制器，其内核采用了 32 位的 ARM Cortex-M4 下内核，最高可支持的主频为 80MHz，256KB 内部闪存、32KB SRAM、多种串口通信。其结构框图如图 4.1.4 所示。

飞行控制板(如图 4.1.5 所示)通过电机驱动程序调整飞行控制板上输出的四路 PWM 波占空比，以调节 4 个电机转速，实现对四旋翼飞行器运动状态的控制；连接超声波、光流、六轴传感器获取姿态信息，在飞行控制板上实时运行各种算法，对飞行器进行高度控制、位姿调节等稳定控制，并与上位机、地面站建立实时的交互数据通信链路，实现四旋翼飞行器的定高、悬停以及完成各种飞行动作指令。

地面控制板采用了意法半导体集团开发的 STM32 微处理器，STM32 基于 ARM Cortex-M3 内核，有着极高的性价比、多样化的产品线、简易的开发方式等优点。按性能分成两个不同的系列，即 STM32F103“增强型”系列和 STM32F101“基本型”系列。增强型系列时钟频率达到 72MHz，是同类产品中性能最高的产品；基本型时钟频率为 36MHz，以 16 位产品的价格得到比 16 位产品大幅提升的性能，是 16 位产品用户的最佳选择。两个系列都内置 32KB 到 128KB 的闪存，不同的是 SRAM 的容量和外设接口的组合。时钟频率为 72MHz 时，从闪存执行代码，STM32 功耗 36mA，是 32 位市场上功

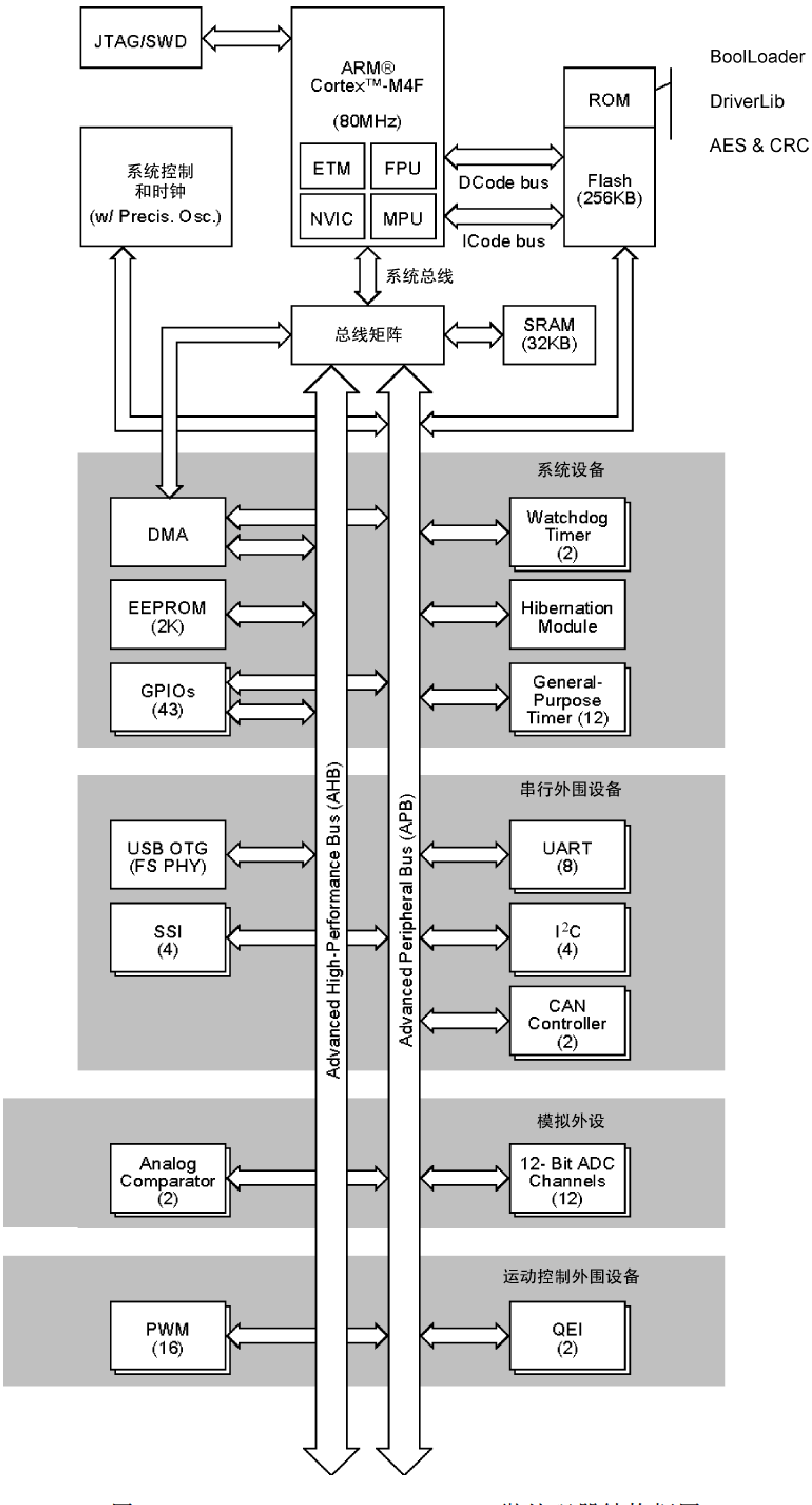


图 4.1.4 Tiva TM4C123G H6PM 微处理器结构框图

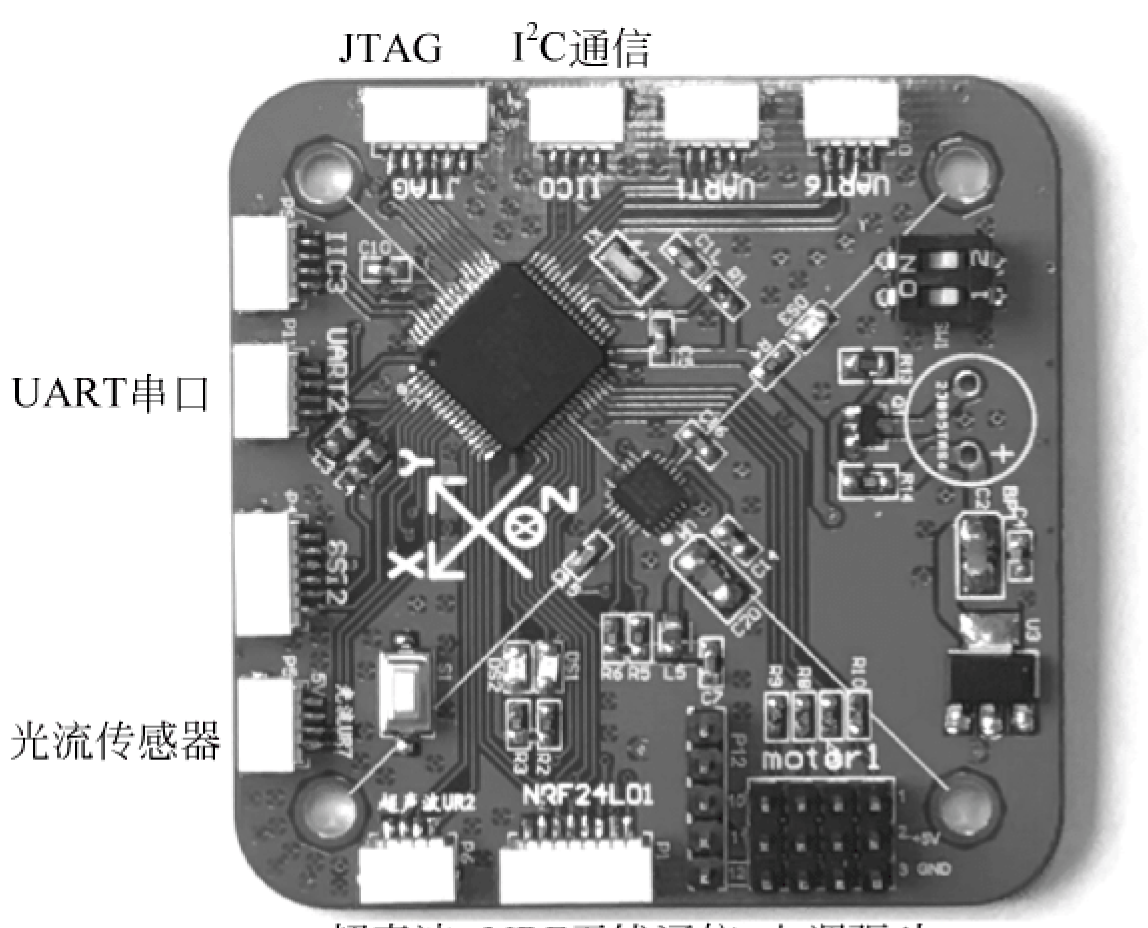


图 4.1.5　飞行控制板实物

耗最低的产品，相当于 0.5mA/MHz。STM32 是基于固件库的开发方式，只需要调用应用程序接口 API 而不需要直接对底层寄存器操作就可以完成应用程序的开发，从而缩短了程序开发周期。STM32F103 增强型单片机有以下性能。

① 内核：ARM32 位 Cortex-M3 CPU，最高工作频率 72MHz，1.25DMIPS/MHz。单周期乘法和硬件除法。

② 存储器：片上集成 32～512KB 的 Flash 存储器以及 6～64KB 的 SRAM 存储器。

③ 时钟、复位和电源管理：2.0～3.6V 的电源供电和 I/O 接口的驱动电压，POR、PDR 和可编程的电压探测器(PVD)，4～16MHz 的晶振，内嵌出厂前调校的 8MHz RC 振荡电路。内部 40kHz 的 RC 振荡电路。用于 CPU 时钟的 PLL。带校准用于 RTC 的 32kHz 的晶振。

④ 低功耗：3 种低功耗模式，即休眠、停止、待机模式，为 RTC 和备份寄存器供电的 VBAT。

⑤ 调试模式：串行调试(SWD)和 JTAG 接口。

⑥ 最多高达 112 个的快速 I/O 端口：根据型号的不同，有 26、37、51、80 和 112 的 I/O 端口，所有的端口都可以映射到 16 个外部中断向量。除了模拟输入，所有的都可以接收 5V 以内的输入。最多达 11 个定时器：4 个 16 位定时器，每个定时器有 4 个 IC/OC/PWM 或者脉冲计数器；2 个 16 位的 6 通道高级控制定时器，最多 6 个通道可用于 PWM 输出；2 个看门狗定时器(独立看门狗和窗口看门狗)；Systick 定时器，24 位倒计

数器；2 个 16 位基本定时器，用于驱动 DAC。最多达 13 个通信接口：2 个 IIC 接口(SMBus/PMBus)；5 个 USART 接口(ISO7816 接口，LIN，IrDA 兼容，调试控制)；3 个 SPI 接口(18Mbit/s)，2 个和 IIS 复用；CAN 接口(2.0B)；USB 2.0 全速接口；SDIO 接口。

⑦ ECOPACK 封装：STM32F103xx 系列微控制器采用 ECOPACK 封装形式。

以 STM32F103RBT 为例，STM32F103RBT6 单片机核心板电路主要包括电源滤波电路、晶振电路、复位电路和启动配置电路，在层次原理图中产生的子模块的原理图文件中完成这些功能电路的绘制，绘制好的电路如图 4.1.6 所示。

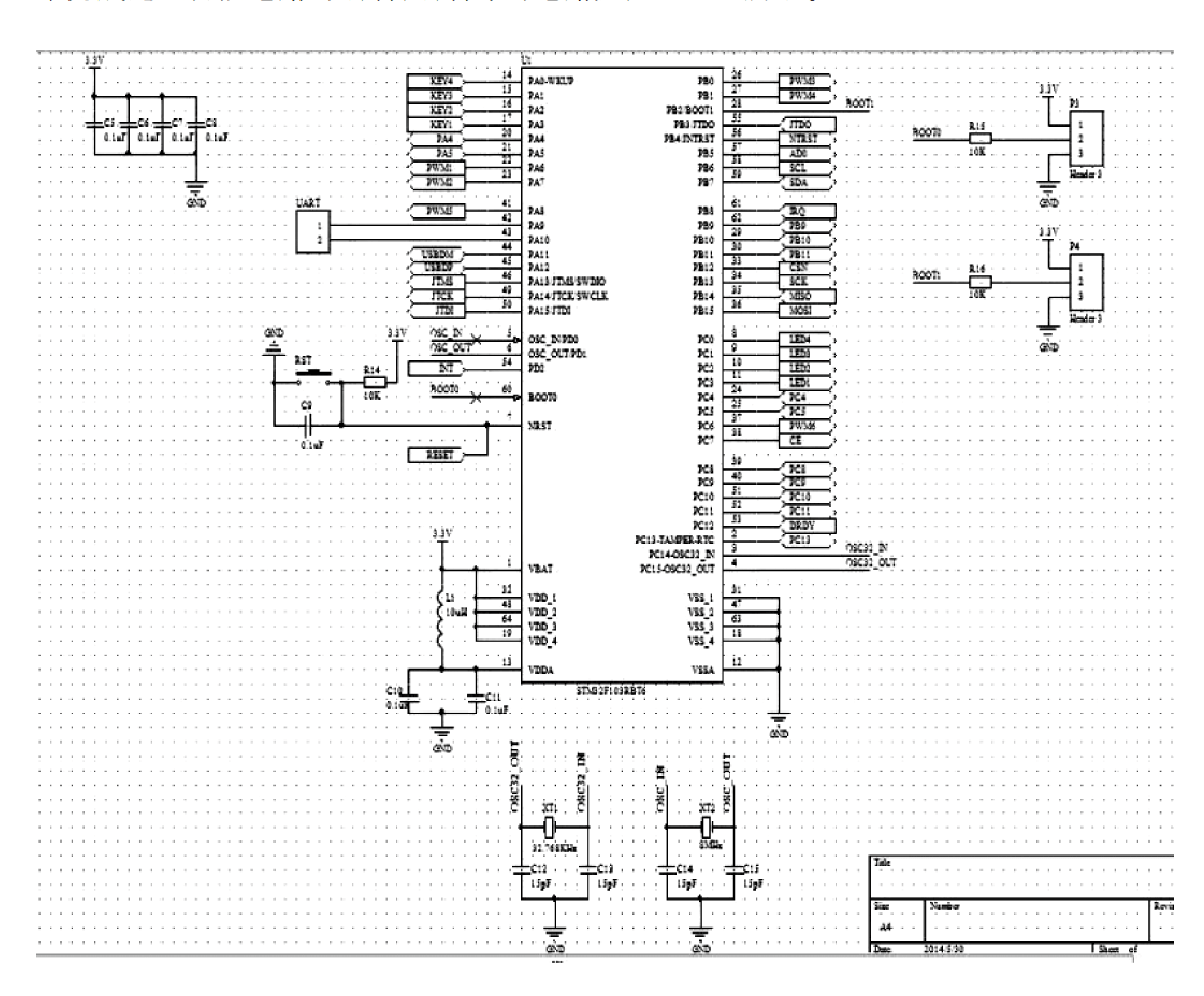


图 4.1.6 STM32F103RBT6 单片机核心板电路

为了保证单片机工作时要求的稳定电压，电路中最好要有电源滤波电路，本设计里用了数个 0.1μF 的电容作为滤波电容，对于单片机需要接电源的端口需要注意以下

两点。

① 无论是否使用模拟部分和 AD 部分，MCU 外围为 VCC、GND、VDDA、VSSA、Vref（如果封装有该引脚）都必须连接，不可悬空。

② 对于每组对应的 V_{CC} 和 GND 都应至少放置一个 0.1μF 的陶瓷电容用于滤波，并且该电容放置应尽量靠近 MCU。

由于单片机所接的外设很多，所以必须合理分配引脚功能，以便充分利用单片机的内部资源，STM32 系列单片机制造商意法半导体为此开发了一个软件 MicroXplorer，使用这个工具能很直观地对端口进行分配，分配好的 IO 资源如图 4.1.7 所示。

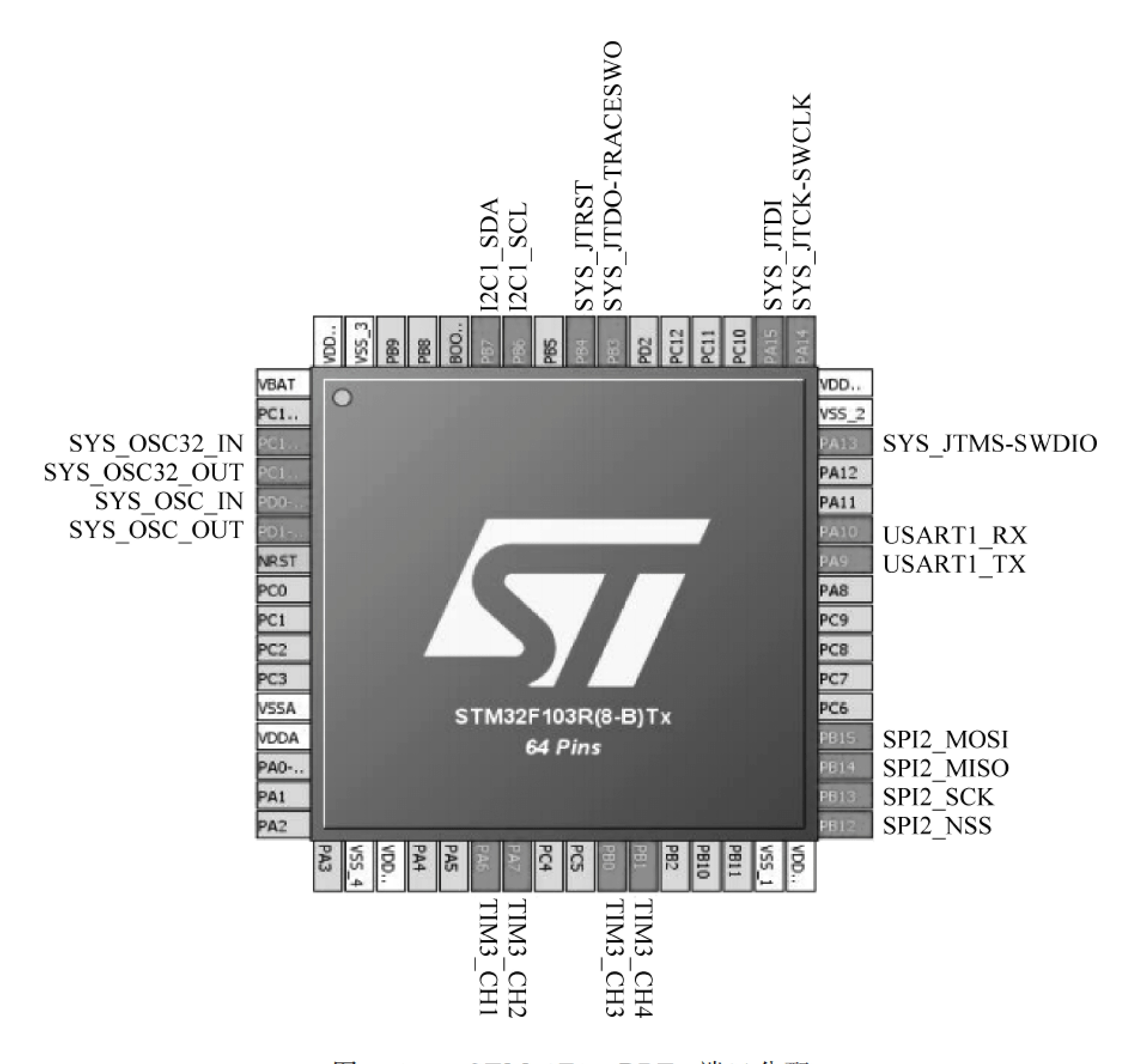


图 4.1.7 STM32F103RBT6 端口分配

2. 飞行姿态感测模块

陀螺仪是测定物体旋转角度的传感器，能测定整个飞行器的姿态。最早的陀螺仪是机械式的，近年来随着 MEMS（微机电）技术的发展，陀螺仪已经做得很小，且可支持 I^2C、

SPI 等总线，大大方便了数据的读取和 PCB 设计，以下介绍 Invensense 公司生产的一款 MPU6050 陀螺仪。

MPU-6050 是一个只有 4mm×4mm×0.9mm 的六轴运动处理传感器(如图 4.1.8 所示)，以数字输出六轴的旋转矩阵、欧拉角格式、四元数的融合演算数据，集成了 3 轴 MEMS 陀螺仪、3 轴 MEMS 加速度计、1 个可以扩展的数字运动处理器(Digital Motion Processor，DMP)。DMP 有权使用 MPU 的一个外部引脚产生中断，可用 I²C 接口连接一个第三方的数字传感器(如磁力计)。扩展之后，它可以利用 I²C 或 SPI 接口输出一个 9 轴的信号。

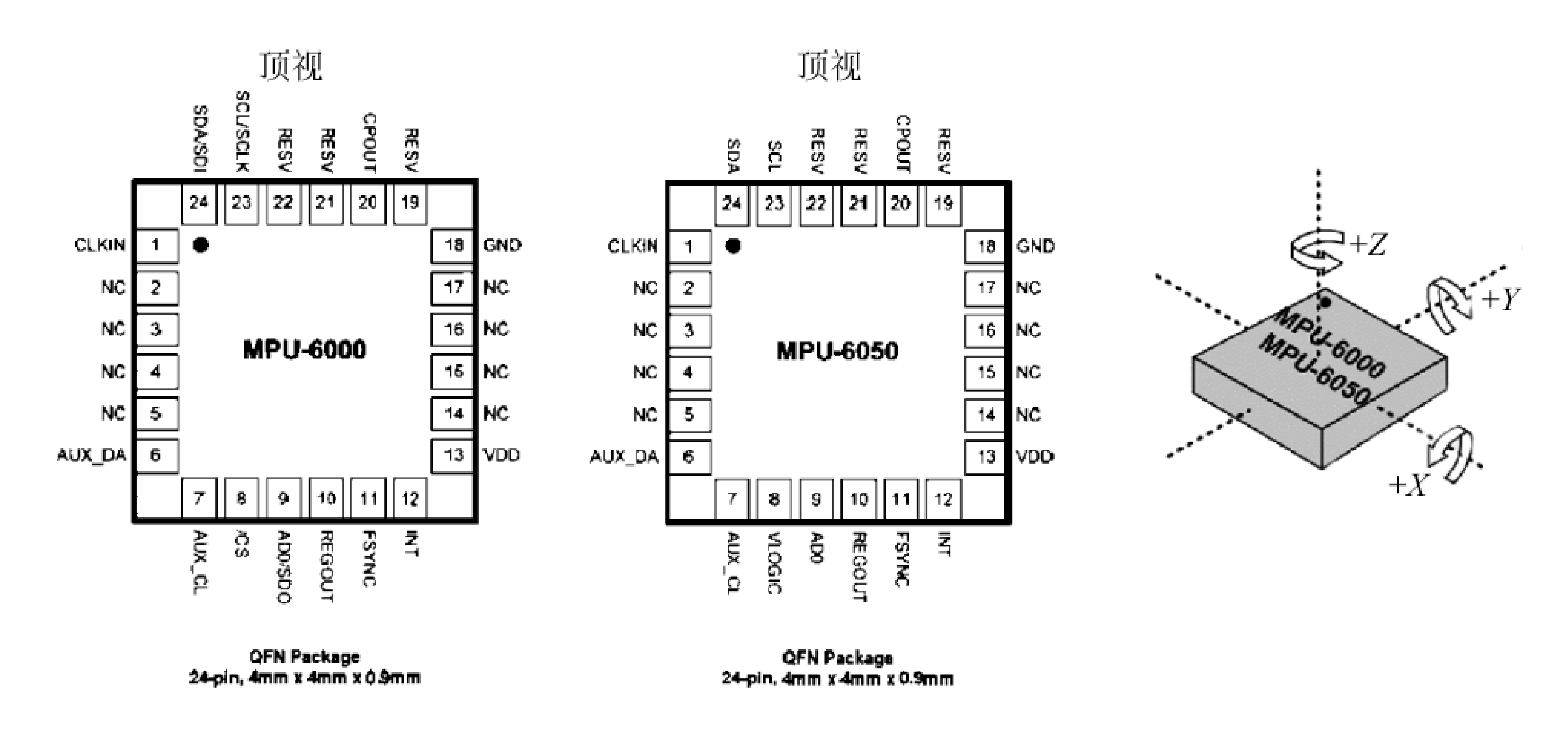


图 4.1.8 MPU-6050

MPU-6050 对加速度计和陀螺仪分别采用 3 个 16 位的 ADC 模块，将测量出的模拟量转化成数字量输出。其引脚配置如表 4.1.1 所示。用户可以控制传感器的测量范围，其中陀螺仪性能参数如下。

表 4.1.1 MPU6050 引脚说明

引脚编号	MPU-6000	MPU-6050	引脚名称	描　述
1	Y	Y	CLKIN	可选的外部时钟输入，如果不用则连到 GND
6	Y	Y	AUX_DA	I²C 主串行数据，用于外接传感器
7	Y	Y	AUX_CL	I²C 主串行时钟，用于外接传感器
8	Y		/CS	SPI 片选(0=SPI mode)
8		Y	VLOGIC	数字 I/O 供电电压
9	Y		AD0/SDO	I²C Slave 地址 LSB(AD0)；SPI 串行数据输出(SDO)

续表

引脚编号	MPU-6000	MPU-6050	引脚名称	描　述
9		Y	AD0	I^2C Slave 地址 LSB(AD0)
10	Y	Y	REGOUT	校准滤波电容连线
11	Y	Y	FSYNC	帧同步数字输入
12	Y	Y	INT	中断数字输出(推挽或开漏)
13	Y	Y	VDD	电源电压及数字 I/O 供电电压
18	Y	Y	GND	电源地
19,21,22	Y	Y	RESV	预备,不接
20	Y	Y	CPOUT	电荷泵电容连线
23	Y		SCL/SCLK	I^2C 串行时钟(SCL);SPI 串行时钟(SCLK)
23		Y	SCL	I^2C 串行时钟(SCL)
24	Y		SDA/SDI	I^2C 串行数据(SDA);SPI 串行数据输入(SDI)
24		Y	SDA	I^2C 串行数据(SDA)
2,3,4,5,14,15,16,17	Y	Y	NC	不接

① 电压:3～6V。

② 电流:<10mA。

③ 体积:15.24mm×15.24mm×2mm。

④ 焊盘间距:上下 100mil(2.54mm),左右 600mil(15.24mm)。

⑤ 测量维度:加速度、角速度及姿态角均三维。

⑥ 量程:加速度为±16g,角速度为±2000°/s。

⑦ 分辨率:加速度为 6.1$e^{-5}g$,角速度为 7.6e^{-3}°/s。

⑧ 稳定性:加速度为 0.01g,角速度为 0.05°/s。

⑨ 姿态测量稳定度:0.01°。

⑩ 数据输出频率 100Hz(波特率 115 200)/20Hz(波特率 9600)。

⑪ 数据接口:串口(TTL 电平),I^2C(直接连 MPU6050,无姿态输出)。

此类数字传感器可以看成一个小型的单片机,数据存储在其内部的寄存器。在使用时,首先通过 I^2C 总线对其控制寄存器写入设置位数据,然后通过 I^2C 总线从存储三轴旋转角速度的寄存器中读回数据即可。这里读回的数据可以是 MPU6050 的 ADC 传回的原始数据,经过转化为四元数和欧拉角后可以用来修正飞行器的飞行姿态,也可以读取 MPU6050 内部集成的数字运动处理器(DMP)所得出的姿态解算结果,从而单片机只需等待 DMP 解算完成后产生的外部中断,使得单片机能有大量的时间来处理其他任务,提

高了系统的实时性。

在具体通信中，需要使用 I^2C 总线来与 MPU6050 进行数据交互。得到了 MPU6050 的传回数据之后，就能够对数据进行融合，得到稳定、可靠的姿态数据，可用于飞行器的平稳飞行控制。

3. 电子调速器

电子调速器在航模中，由于无刷电机在同功率下体积要比有刷的小，所以一般指的是“无刷电机电子调速器”，通常由 MOS 管配合单片机组成，其主要作用是把航模电池提供的直流电转换成三相驱动电流，并通过过零检测等电路检测无刷电机的反馈，从而顺利地驱动无刷电机。

现在的航模电调品牌国外以美国的凤凰最为出名，国内主要品牌有好盈、中特威、飞盈佳乐等。选择合适的航模电调需要根据用户的应用情况和电机的功率要求来选择。选择电调一定要看该款电调的功率，另外要看电调与电机的兼容度。电调并不能兼容所有电机，它必须根据电机的功率等参数来进行选择。但实际情况是许多品牌的电调并不能足功率足电流输出，如 60A 的电调，其实只能跑到 55A 就无法再往上调了，所以选择电调的时候一定要问清楚供应商是否足功率。选择低于电机功率的航模电调会导致电调板上功率管烧坏。所以功率问题是航模电调选择的重要问题。

无刷电机电子调速器主要特性有以下几点。

① 采用功能强大、高性能 MCU 处理器，用户可以针对自身需求设置使用功能，充分体现系统独具优势的智能特点。

② 支持无刷电机无限制最高转速。

③ 支持定速功能。

④ 精心的电路设计，抗干扰性超强。

⑤ 启动方式可设置，油门响应速度快，并具有非常平稳的调速线性，兼容固定翼飞机及直升飞机。

⑥ 低压保护阈值可设置。

⑦ 内置 SBEC，带舵机负载功率大、功耗小。

⑧ 具备多种保护功能。输入电压异常保护/电池低压保护/过热保护/油门信号丢失降功率保护。

⑨ 通电安全性能好：接通电源时无论遥控器油门拉杆在何位置都不会立即启动电机。

⑩ 过温保护。电子调速器工作中当温度到达100℃时功率输出会自动降低一半,低于100℃时功率输出自动恢复。

⑪ 兼容所有遥控器操作设置和支持编程卡设置。

⑫ 设置报警音,判断通电后工作情况。

电子调速器有许多不同的规格,表4.1.2对其中部分进行了列举。

表4.1.2　电调产品规格

规格	持续电流	最大瞬时电流(10s)	BEC输出	BEC模式	电池节数/锂电Li-xx	电池节数/镍氢Ni-xx	尺寸(宽×长×高)/mm	质量/g
FLY-10A	10A	12A	5V/3A-1A	SBEC/UBEC	2-4Lipo	5-12NC	60×26×9	26
FLY-10A OPTO	10A	12A	无	无	2-4Lipo	5-12NC	60×26×9	23
FLY-20A	20A	25A	5V/3A-1A	SBEC/UBEC	2-4Lipo	5-12NC	54×26×11	30
FLY-20A OPTO	20A	25A	无	无	2-4Lipo	5-12NC	54×26×11	26
FLY-30A	30A	40A	5V/3A-1A	SBEC/UBEC	2-4Lipo	5-12NC	54×26×11	32
FLY-30A OPTO	30A	40A	无	无	2-4Lipo	5-12NC	54×26×11	28
FLY-40A	40A	60A	5.5V/4A	SBEC	2-5Lipo	5-15NC	35×71×10.5	48
FLY-40A OPTO	40A	60A	无	无	2-5Lipo	5-15NC	35×71×10.5	42
FLY-50A	50A	80A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×71×17	68
FLY-50A OPTO	50A	80A	无	无	2-6Lipo	5-18NC	35×71×17	62
FLY-60A	60A	80A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×71×17	68
FLY-60A OPTO	60A	80A	无	无	2-6Lipo	5-18NC	35×71×17	62
FLY-70A	70A	100A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×71×17	69
FLY-70A OPTO	70A	100A	无	无	2-6Lipo	5-18NC	35×71×17	63
FLY-80A	80A	100A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×66×22	90
FLY-80A OPTO	80A	100A	无	无	2-6Lipo	5-18NC	35×66×22	82
FLY-90A	90A	120A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×66×22	95
FLY-90A OPTO	90A	120A	无	无	2-6Lipo	5-18NC	35×66×22	87
FLY-100A	100A	140A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×71×22	95
FLY-100A OPTO	100A	140A	无	无	2-6Lipo	5-18NC	35×71×22	87
FLY-110A	110A	150A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×71×22	95
FLY-110A OPTO	110A	150A	无	无	2-6Lipo	5-18NC	35×71×22	87
FLY-120A	120A	150A	5.5V/4A	SBEC	2-6Lipo	5-18NC	35×71×22	98
FLY-120A OPTO	120A	150A	无	无	2-6Lipo	5-18NC	35×71×22	90

电子调速器的接线如图 4.1.9 所示。

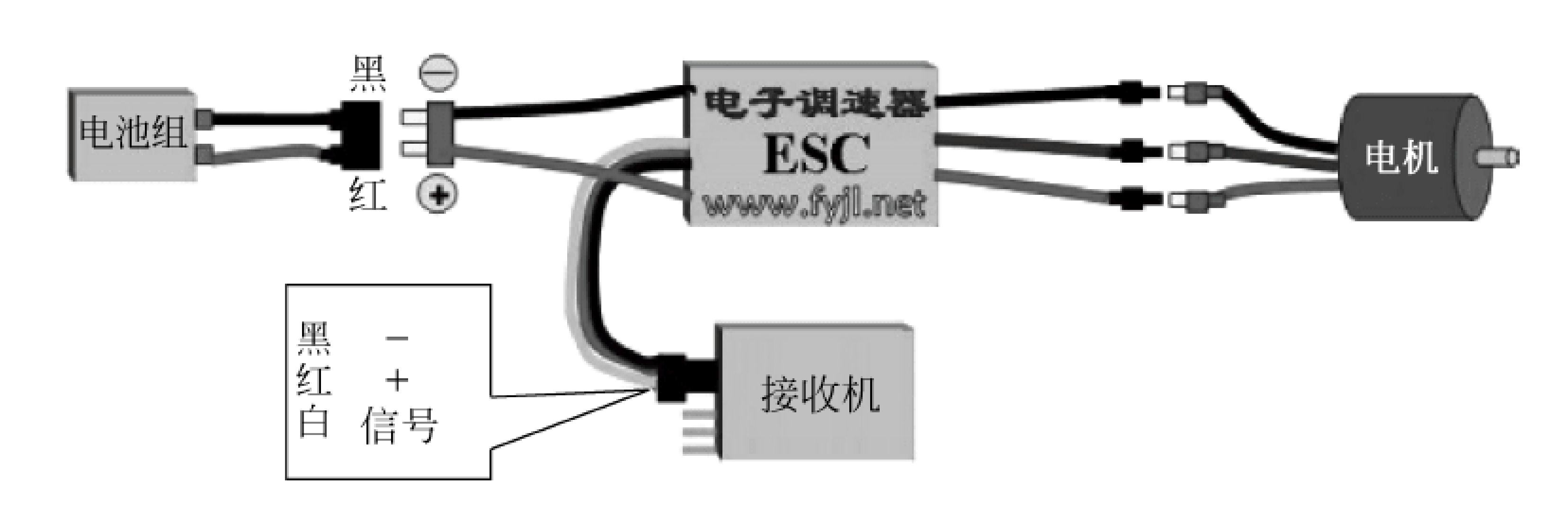


图 4.1.9 电子调速器连接线说明(为避免短路和漏电,连接处均使用热缩导管绝缘)

不同的电调有不同的设置指令(通常是从信号引脚输入不同占空比的 PWM 波),飞行控制接收机或者其他能发出 50Hz、5%~10%的 PWM 波设备(信号源、遥控器、发射机等)均可以对电调进行各种设置。需要特别说明的是,一般而言,电调的 BEC 输出是 5V,但电调的控制信号并不需要 5V,只是 3.3V 就可以识别了。这是因为 5V 类型电平(CMOS 电平)的高电平识别下限是 2.7V,而目前常用的 3.3V 单片机输出的高电平已经超过了这个下限,因此在使用此类单片机调试电调的时候不用刻意去设计电平转换电路。

从说明书可以获得电子调速器的设置方法,其中最重要的是油门行程的设置。油门行程是指当信号接收引脚收到的 PWM 波的占空比变化 1%时,电调控制电机转速对应进行多少变化。4 个电调必须有同样的油门行程,这样当飞行控制控制电调从而控制电机转速时,对应的转速变化才比较好控制。

油门行程的设置方法(当前大多兼容电调程序)如图 4.1.10 所示。

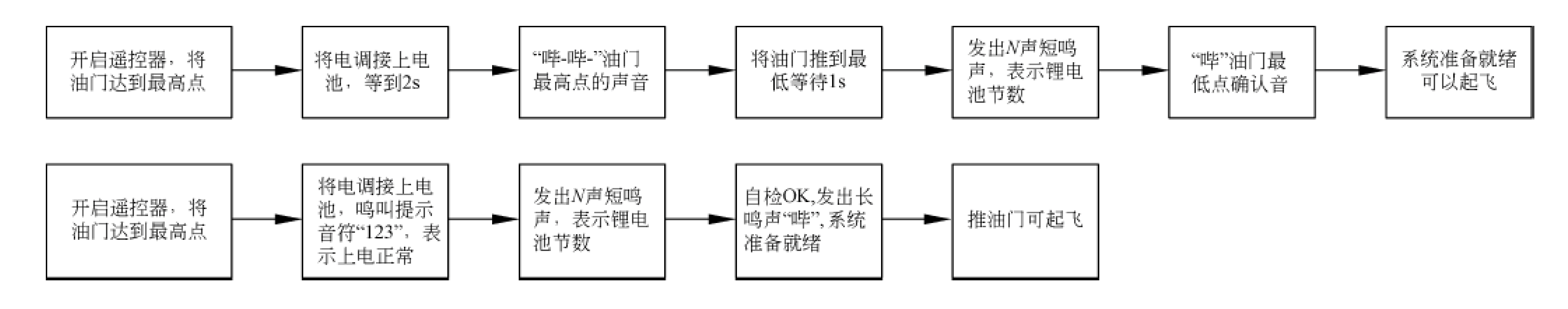


图 4.1.10 电调油门行程校准过程

这一过程从信号变化来看,就是保持 10%左右的占空比—等待 2s—等待确认音—保持 5%左右占空比—等待 1s—确认锂电池和油门。也就是一个让电调确定收到的 PWM 波占空比的范围,从而能够根据这个范围来确认,当占空比变化时,自己应该怎么控制电机转速做出相应变化的过程。

4. 无线通信模块

2.4GHz ISM(Industry Science Medicine)是全世界公开通用的无线频段,被目前广泛应用于家用及商用领域,它是一个全球性的频段,开发的产品具有全球通用性,各类无线产品均可使用此频段来实现无线通信的功能。下面介绍一种基于该频段的无线通信模块——nRF2401。

nRF2401 支持多点间通信,最高传输速率超过 1Mb/s,而且比蓝牙具有更高的传输速度。它采用 SoC 方法设计,只需少量外围元件便可组成射频收发电路。与蓝牙不同的是,nRF2401 没有复杂的通信协议,它完全对用户透明,同种产品之间可以自由通信。更重要的是,nRF2401 比蓝牙产品更便宜。所以 nRF2401 是业界体积最小、功耗最少、外围元件最少的低成本射频系统级芯片。nRF2401 内置地址解码器、先入先出堆栈区、解调处理器、时钟处理器、GFSK 滤波器、低噪声放大器、频率合成器及功率放大器等功能模块,只需要很少的外围元件,使用起来非常方便。QFN24 引脚封装,外形尺寸只有 5mm×5mm。

nRF2401 的功能模块如图 4.1.11 所示,引脚排列如图 4.1.12 所示,引脚功能如表 4.1.3 所示。

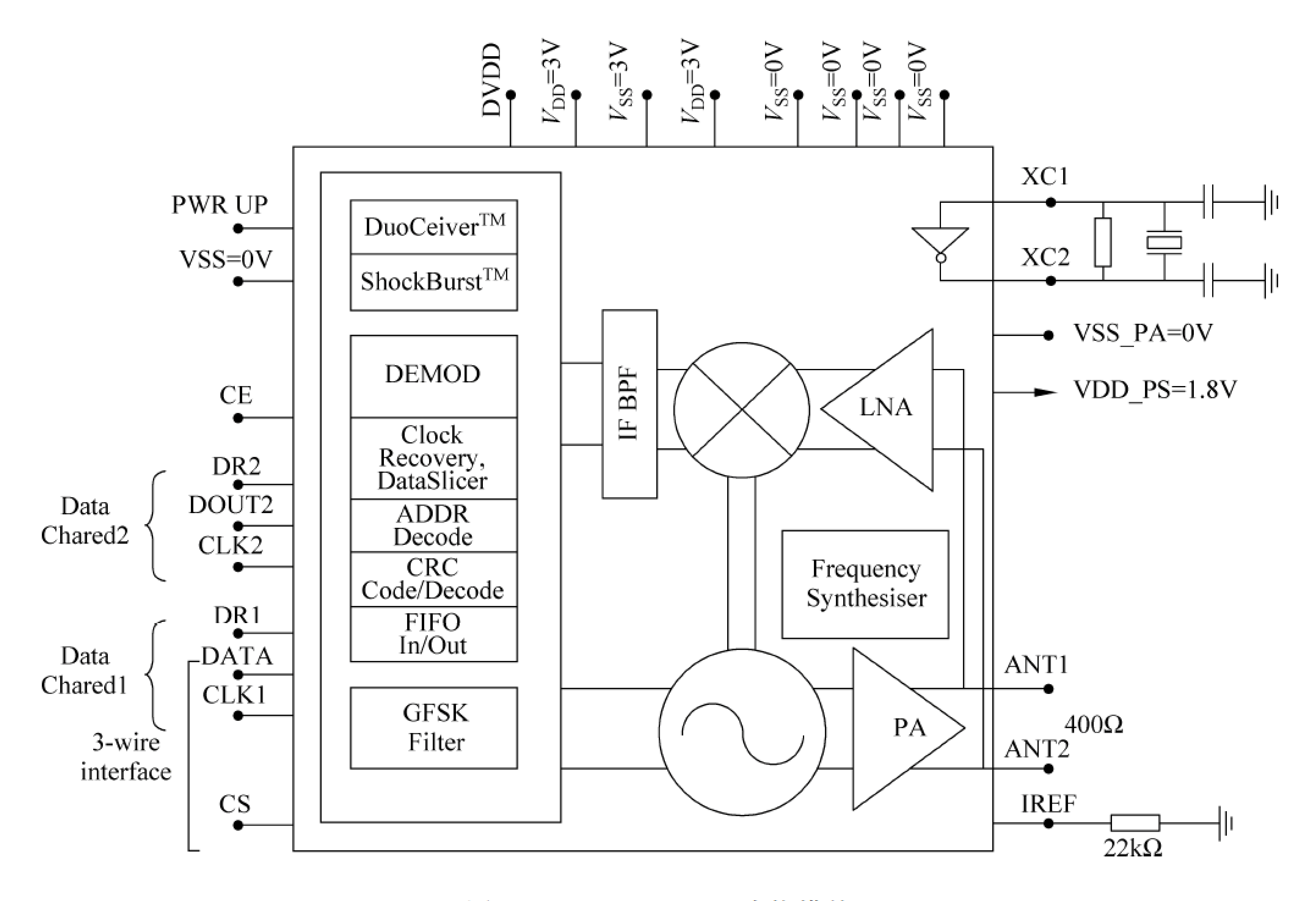


图 4.1.11　nRF2401 功能模块

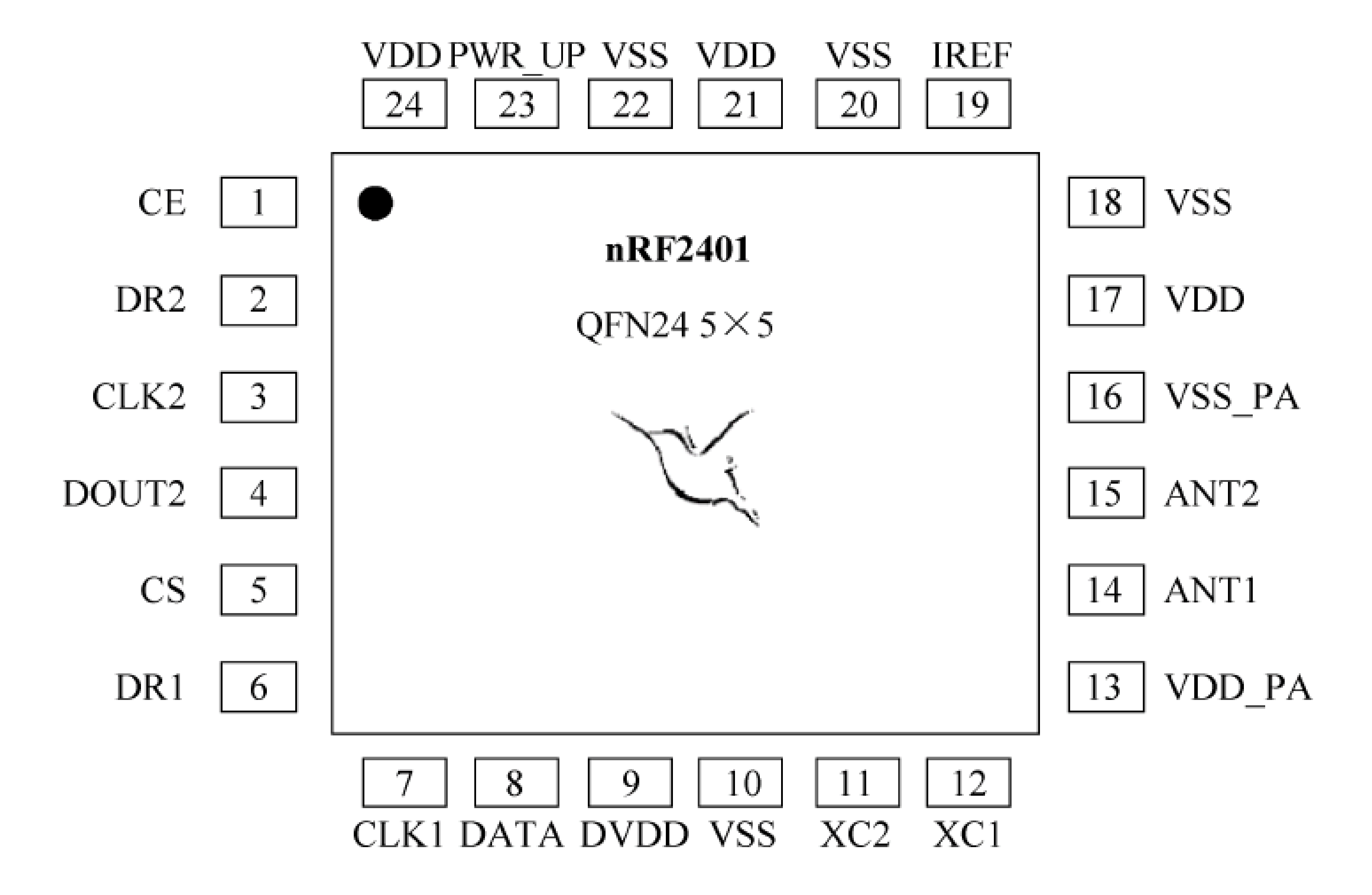


图 4.1.12 nRF2401 引脚排列

nRF2401 的主要特点如下。

① 采用全球开放的 2.4GHz 频段，有 125 个频道，可满足多频及跳频需要。

② 速率(1Mb/s)高于蓝牙，且具有高数据吞吐量。

③ 外围元件极少，只需一个晶振和一个电阻即可设计射频电路。

④ 发射功率和工作频率等所有工作参数可全部通过软件设置。

⑤ 每个芯片均可以通过软件设置最多 40 位地址，而且只有收到本机地址时才会输出数据(提供一个中断指示)，同时编程也很方便。

⑥ 采用 ShockBurstlM 模式时，能适用极低的功率操作和不严格的 MCU 执行。

⑦ 带有集成增强型 8051 内核、9 路 10 位 ADC、UART 异步串口、SPI 串口和 PWM 输出。

⑧ 内置看门狗。

表 4.1.3 nRF2401 各引脚功能说明

引脚	名称	引脚功能	描　述
1	CE	数字输入	使 nRF2401 工作于接收或发送状态
2	DR2	数字输出	频道 2 接收数据准备好
3	CLK2	数字 I/O	频道 2 接收数据时钟输入输出
4	DOUT2	数字输出	频道 2 接收数据
5	CS	数字输入	配置模式的片选端
6	DR1	数字输出	频道 1 接收数据准备好
7	CLK1	数字 I/O	频道 1 接收数据时钟输入输出
8	DATA	数字 I/O	频道 1 接收发送数据端

续表

引脚	名称	引脚功能	描　　述
9	DVDD	电源	电源的正数字输出
10	VSS	电源	电源地
11	XC1	模拟输出	晶振 1
12	XC2	模拟输入	晶振 2
13	VDD_PA	电源输出	给功率放大器提供 1.8V 的电压
14	ANT1	天线	天线接口 1
15	ANT2	天线	天线接口 2
16	VSS_PA	电源	电源地
17	VDD_PA	电源	电源正端
18	VSS	电源	电源地
19	IREF	模拟输入	模数转换的外部参考电压
20	VSS	电源	电源地
21	VDD_PA	电源	电源正端
22	VSS	电源	电源地
23	PWR_UP	数字输入	芯片激活端
24	VDD	电源	电源正端

nRF2401 有 4 种工作模式(如表 4.1.4 所示),即收发模式、配置模式、空闲模式和关机模式,由 PWR_UP、CE、和 CS 这 3 个引脚决定。

表 4.1.4　nRF2401 工作模式

工作模式	PWR_UP	CE	CS
收发模式	1	1	0
配置模式	1	0	1
空闲模式	1	0	0
关机模式	0	X	X

收发模式：nRF2401 的收发模式有 ShockBurstTM 收发模式和直接收发模式两种,收发模式由器件配置字决定。这里只介绍 ShockBurstTM 收发模式。

ShockBurstTM 收发模式下,使用片内的先入先出堆栈区,数据低速从微控制器送入而高速(1Mb/s)发射,这样可以尽量节能,因此,使用低速的微控制器也能得到很高的射频数据发射速率。与射频协议相关的所有高速信号处理都在片内进行,这种做法有三大好处：尽量节能；低的系统费用(低速微处理器也能进行高速射频发射)；数据在空中停留时间短,抗干扰能力强。nRF2401 的 ShockBurstTM 技术同时也减小了整个系统的平均工作电流。

在 ShockBurstTM 收发模式下，nRF2401 自动处理字头和 CRC 校验码。在接收数据时，自动把字头和 CRC 校验码移去。在发送数据时，自动加上字头和 CRC 校验码，当发送过程完成后，数据准备好引脚通知微处理器数据发射完毕。

配置模式：在配置模式，15B 的配置字被送到 nRF2401，这需通过 CS、CLK1 和 DATA 3 个引脚完成。

空闲模式：nRF2401 的空闲模式是为了减小平均工作电流而设计，其最大优点是，实现节能的同时缩短芯片的启动时间。在空闲模式下，部分片内晶振仍在工作，此时的工作电流跟外部晶振的频率有关，如外部晶振为 4MHz 时工作电流为 12μA，外部晶振为 16MHz 时工作电流为 32μA。在空闲模式下，配置字的内容保持在 nRF2401 片内。

关机模式：在关机模式下，为了得到最小的工作电流，一般此时的工作电流小于 1μA。关机模式下，配置字的内容也会被保持在 nRF2401 片内，这是该模式与断电状态最大的区别。

5. 电源模块

电源电路作为整个系统的动力源泉必须要设计合理；否则会造成系统工作不稳定。对于地面站系统，STM32F103RBT6 使用 3.3V 电源作为系统电源，而通常的外部电源为 5V，所以必须要通过电源转换电路进行降压稳压处理，如使用 AMS1117 线性稳压芯片，参考 AMS1117 的数据手册并且辅以适当的外围电路便可设计出满足要求的 5V 转 3.3V 的电源电路，如图 4.1.13 所示。

对于飞行器的供电，则需要高性能的航模电池。航模电池常常用几个 S 来代表有几片电池串联，1S 表示一片电池 3.8V，充满电 4.2V。除了 S 数值是选择电池的常用标准外，还有以下指标需要注意。

① 电池的 C 数。一般航模电池会标明电池的 C 数，指的是最大放电倍率。例如，使用的 1S 锂电池，一般是 350mAh、25C。那么这个电池最大可以提供的电流就是 0.35A×25=8.75A。

② 电池的容量。容量越大，存储的能量就越大，可以提供的续航时间就越长，不过相应的重量也越大。较大的四旋翼飞行器常用 2200mAh、3S、25C 的电池。

注意使用对应的充电器。由于航模电池的电流极大，因此一般采用平衡充电器来分别充其中的每一个 S，也就是每一个放电单元，这样比较保护电池。不过由于小四旋翼飞行器的电池只有一个 S，所以可以用比较简单的充电器完成充电。不要过度放电。航模电池如果放电过多，会导致下一次无法使用，电池也会起鼓包。

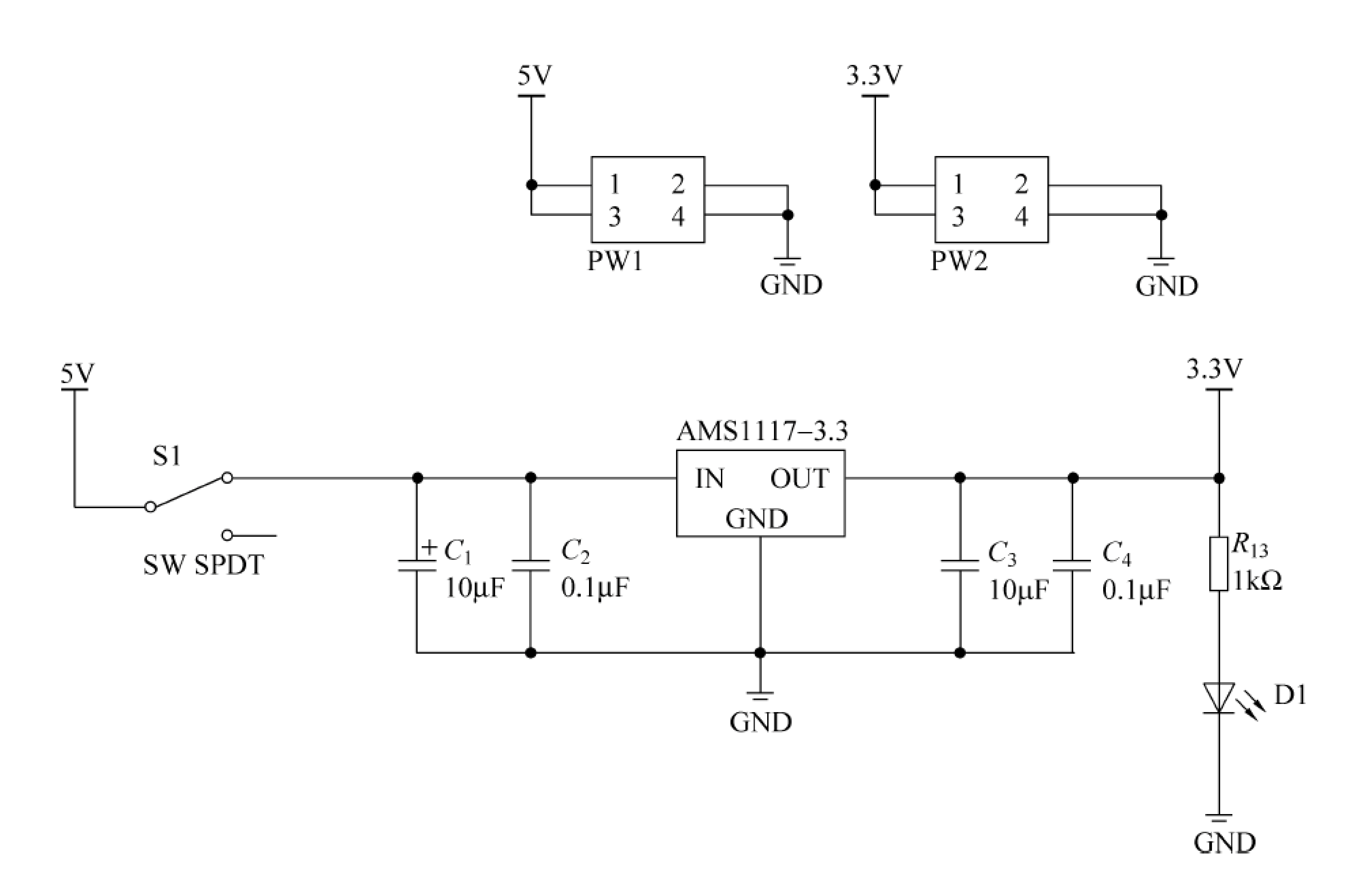


图 4.1.13 线性稳压电源电路

四旋翼飞行器电池还要注意分线板的使用。由于电池只有一个 T 形插口，但是要将电量分给 4 个电调使用，分线板可以将多个电调并联到电池电源上。

最后，电池在使用的过程中，需要注意与机身的固定，以免飞行中掉落。大四旋翼飞行器有专门的绑带出售，小四旋翼飞行器电池可以用透明胶带或者双面胶粘贴。

4.1.2 外围传感器

1. GPS 定位模块

全球定位系统(Global Positioning System，GPS)起始于 1958 年美国军方的一个项目，1964 年投入使用，其基本原理是测量出已知位置的卫星到用户接收机之间的距离，然后综合多颗卫星的数据就可知道接收机的具体位置。四旋翼飞行器在室外飞行时，通过搭载的 GPS 可以获取飞行器当前所处的位置和速度信息。



图 4.1.14 NEO-6 GPS 模块

图 4.1.14 所示为瑞士 U-blox 公司推出的 NEO6 系列 GPS 模块，图 4.1.15 所示为其设计原理图。这块 GPS 模块是 2009 年推出的，降低了

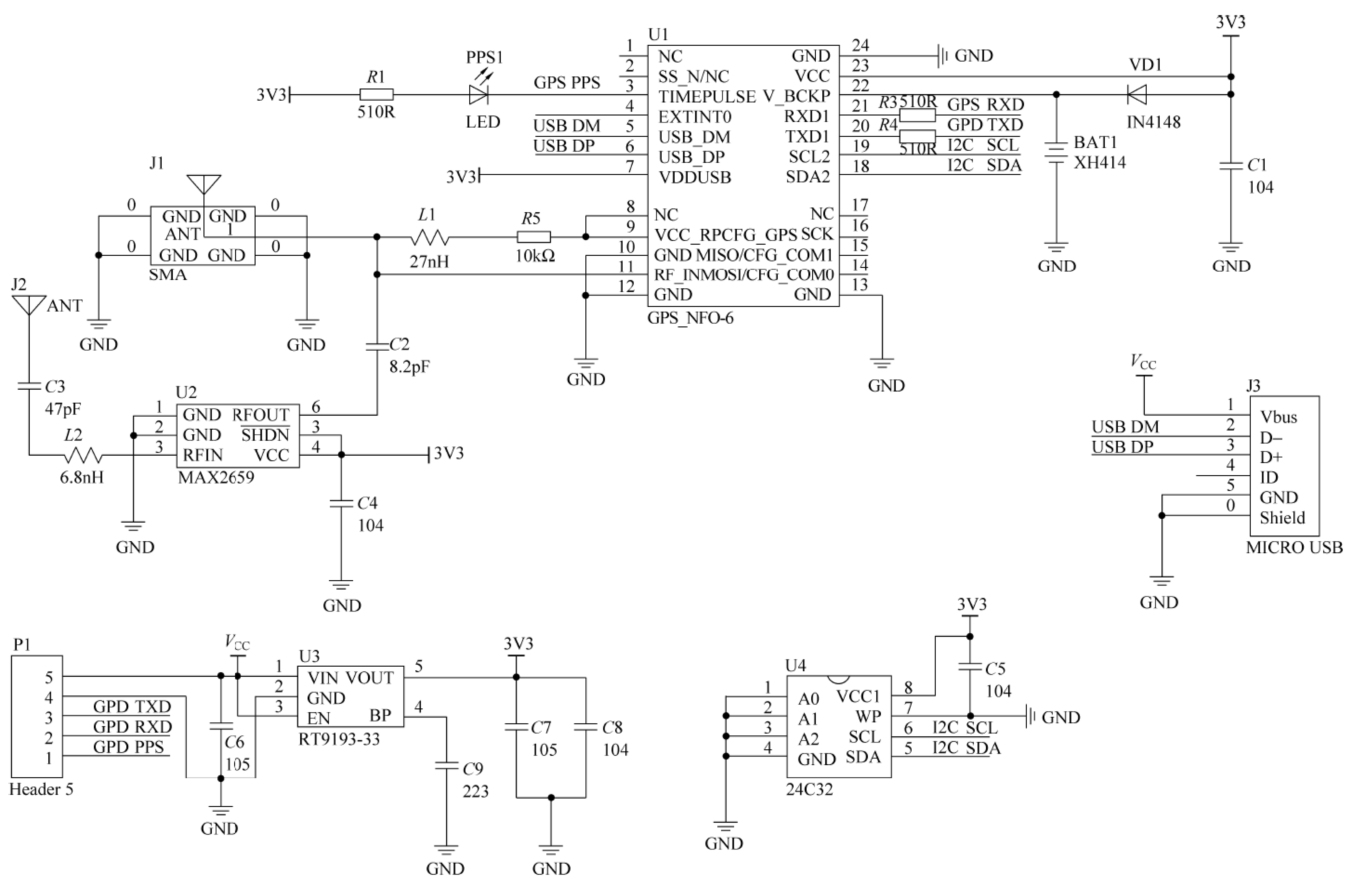


图 4.1.15　NEO6 模块参考设计原理图

GPS 模块的功耗，具备智能化功率管理功能，延长了电池寿命。同时 NEO6 模块中捕获引擎的相关数量拓展到 200 万个以上，在提高微弱信号捕获能力的同时，缩短了首次定位时间。NEO6 具有以下特点。

① UART 接口：支持数据传输速率 4.8～230KB/s，信号输入输出电平为 V_{CC}。

② USB2.0 接口：支持全双工模式，数据传输速率可达 12Mb/s。

③ DDC(Display Data Channel)：兼容 I^2C 接口，NEO6 接收机运行在 I^2C 从模式下。当外部有 EEPROM 时，可以工作在主模式下。

④ SPI 接口：允许 NEO6 通过 SPI 接口连接外部的设备，如 Flash、AD 转换器，或者是主 CPU。

⑤ 丰富的 GPIO 接口。

此外，NEO6 GPS 模块还具有以下特性。

① 模块采用 U-BLOX NEO-6M 模组，体积小巧，性能优异。

② 模块增加放大电路，有利于无缘陶瓷天线快速搜索卫星。

③ 模块可通过串口进行各种参数设置，并可保存在 EEPROM 中，使用方便。

④ 模块自带 SMA 接口，可以连接各种有源天线，适应能力强。

⑤ 模块兼容 3.3V/5V 电平，方便连接各种单片机系统。

⑥ 模块自带可充电后备电池，可以掉电保持星历数据。

2. 磁力计(HMC5883)

霍尼韦尔 HMC5883L 磁强计是一种采用无铅表面封装技术的高集成模块，带有数字接口的弱磁传感器芯片，其包括高分辨率的 HMC118X 系列磁阻传感器、放大器、自动消磁驱动器、偏差校准和数模转换器。

其引脚配置如图 4.1.16 所示。

HMC5883L 磁强计相关参数如下：

① 封装尺寸：3.0mm×3.0mm×0.9mm。

② 测量分辨率：5mGa。

③ 工作电压：2. 16～3.6V。

④ 测量精度：1°～2°。

⑤ 工作电流：100μA。

⑥ 最大输出频率：160Hz。

⑦ 工作温度：－40～＋85℃。

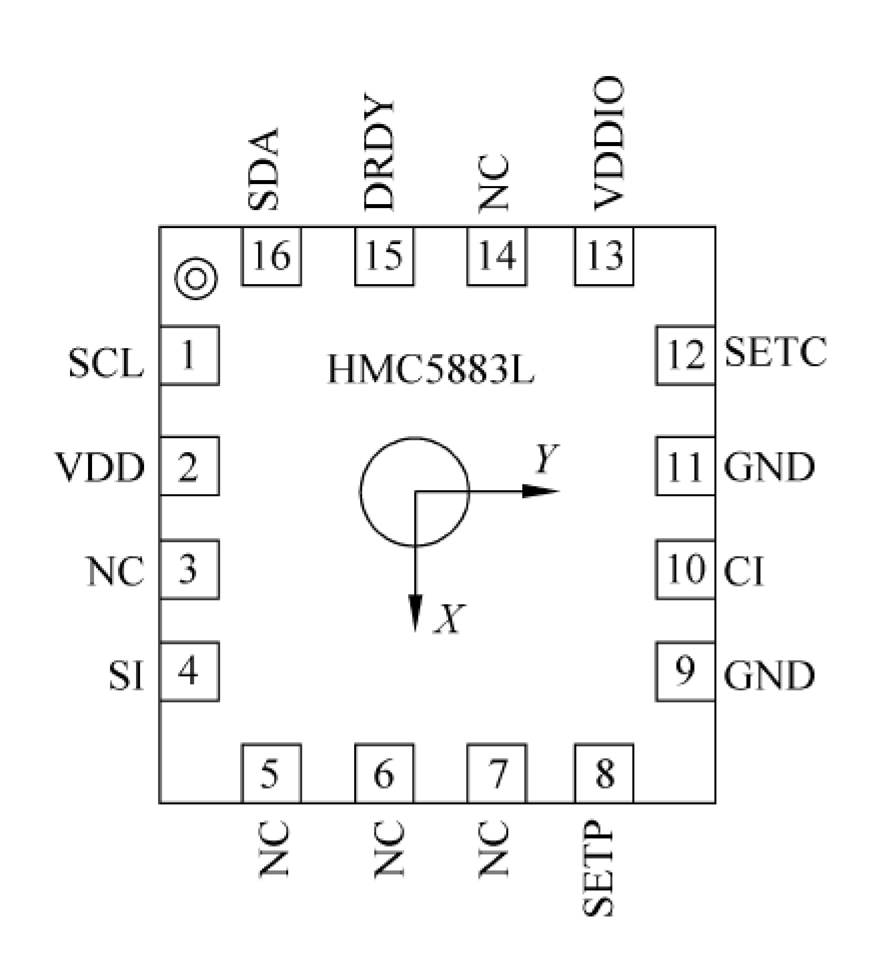


图 4.1.16 HMC5883L 磁强计引脚排列

⑧ 寄存器位数：8 位。

⑨ 测量磁场范围：±8Ga。

⑩ ADC 位数：12 位。

HMC5883L 磁强计采用各向异性磁阻(MR)技术使各向异性传感器具有在轴向高灵敏度和线性高精度的特点，通过施加供电电源，传感器可以将测量轴方向上的任何入射磁场转变成一种差分电压输出，这些磁阻元件两两对齐，形成一个共同的敏感轴(如引脚图上的箭头所示)，随着磁场在敏感轴向上不断增强，电压也就正向增强，该磁强计能够可靠测量地球磁场的方向和大小。HMC5883L 通过总线来控制，外接电阻后，可支持标准模式 l00kHz 和快速模式 400kHz 的时钟频率，但不支持高速模式。其内部时钟具有数字逻辑功能和定时管理功能。HMC5883L 磁强计测向原理是，利用磁强计测量地磁场的水平方向分量，然后确定磁北极和与前进方向的夹角。当磁强计水平放置时，其传感器的坐标轴如图 4.1.17 所示。

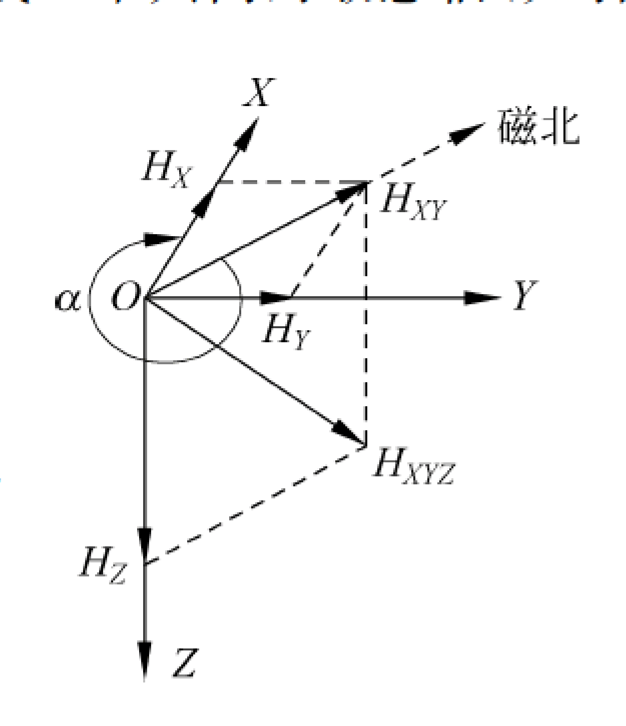


图 4.1.17 磁强计传感器坐标系

X、Y 轴在水平面内，X 轴指向前进方向，Z 轴沿重力方向向下，Y 轴根据右手坐标系确定。从磁北极与轴正方向沿顺时针方向的夹角即为方位角 α。磁北极为地球磁场沿水平方向的分量，而地磁场的北极和地理北极有差别，且处在地球上的位置不同，这种差别也相应不同。所以 HMC5883L 在使用时需要初始化，先补偿当地的磁场大小和方向，即当地的磁偏角，然后才能计算出

磁北极和地北极的差值，从而得出正确的地理北极方向。

实际情况下，当四旋翼飞行器飞行时，磁强计并不水平，此时需要根据坐标变化计算其水平方向分量。

3. 气压计（BMP085）

BMP085 是一款高精度、超低能耗的压力传感器，可以应用在移动设备中。它的性能卓越，绝对精度最低可以达到 0.03hPa，并且耗电极低，只有 3μA。BMP085 采用强大的 8 脚陶瓷无引线芯片承载（LCC）超薄封装，可以通过 I^2C 总线直接与各种微处理器相连。图 4.1.18 是其封装外形和引脚排列。主要特点如下。

① 压力范围：300～1100hPa（海拔 9000～500m）。

② 电源电压：1.8～3.6V（V_{DDA}），1.62～3.6V（V_{DDD}）。

③ LCC8 封装：无铅陶瓷载体封装（LCC）。

④ 尺寸：5.0mm×5.0mm×1.2mm。

⑤ 低功耗：5μA 在标准模式。

⑥ 高精度：低功耗模式下，分辨率为 0.06hPa（0.5m），高线性模式下，分辨率为 0.03hPa（0.25m）。

⑦ 反应时间：7.5ms。

⑧ 待机电流：0.1mA。

⑨ 无须外部时钟电路。

⑩ 含温度输出，IC 接口，温度补偿，无铅，符合 RoHS 规范，MSL 1。

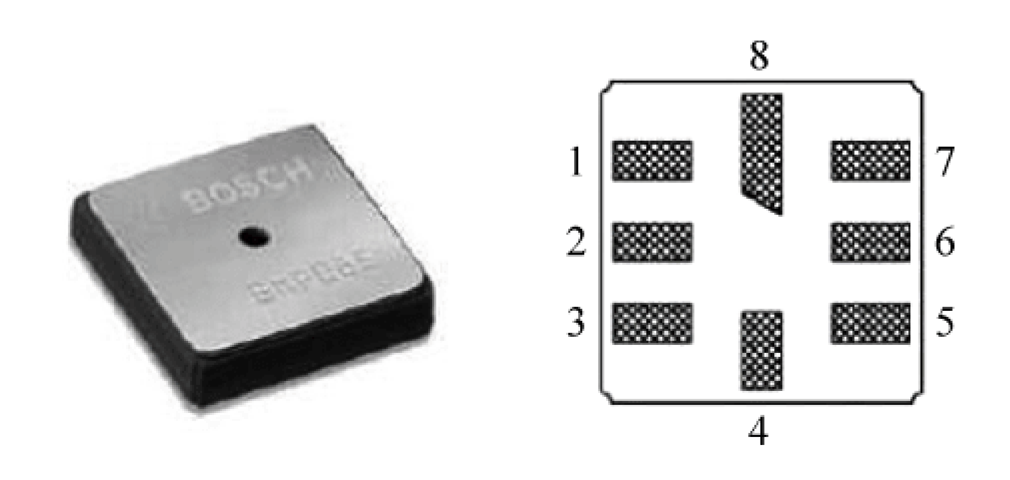


图 4.1.18　封装外形和引脚排列

引脚各功能如下：1 脚（GND）接电源地，2 脚（EOC）为完成转换输出，3 脚（VDDA）为正电源，4 脚（VDDD）为数字正电源，5 脚为空，6 脚（SCL）为 I^2C 总线的时钟端，7 脚（SDA）为 I^2C 总线的数据端，8 脚（XCLR）为主清除信号输入端，低电平有效，用来复位 BMP085 和初始化寄存器和控制器，在不用的情况下可以空置。

4. 超声波模块

由于超声波传感器的成本低、实现方法简单、技术成熟，是无人机常用的传感器。由于对不同材料声波的反射是不同的，且易受到其他声波干扰，以及超声波的测量周期较长，所以超声波传感器一般只用在探测几米之内的距离。

四旋翼无人直升机在飞行高度要求不高，对精度的要求也不是很高的情况下，可选用的超声波为 US-100 或者 KS-103H。模块共有两个接口，即模式选择跳线和 5 针接口。

模式选择跳线的间距为 2.54mm，当插上跳线帽时为 UART(串口)模式，拔掉时为电平触发模式。UART(串口)模式实时性不高，采集的数据不适合用作无人机的定高，因此推荐使用电平触发模式。

5. 光流视觉定位传感器

ADNS-3080 光流传感器价格低廉，容易读取数据，但对地面要求很高。ADNS-3080 是一款光电鼠标传感器，SPI 数据接口，原理图如图 4.1.19 所示。读出的数据需要进行高度修正，即读出来的像素移动值输出结果需要根据高度来标定，为了找到正确的缩放因子，传感器将某个高度作为参考，在不同的高度多次水平移动相同距离，求出缩放因子，使得不同高度移动相同距离变化的像素值相同。具体的软件编程见后面部分。

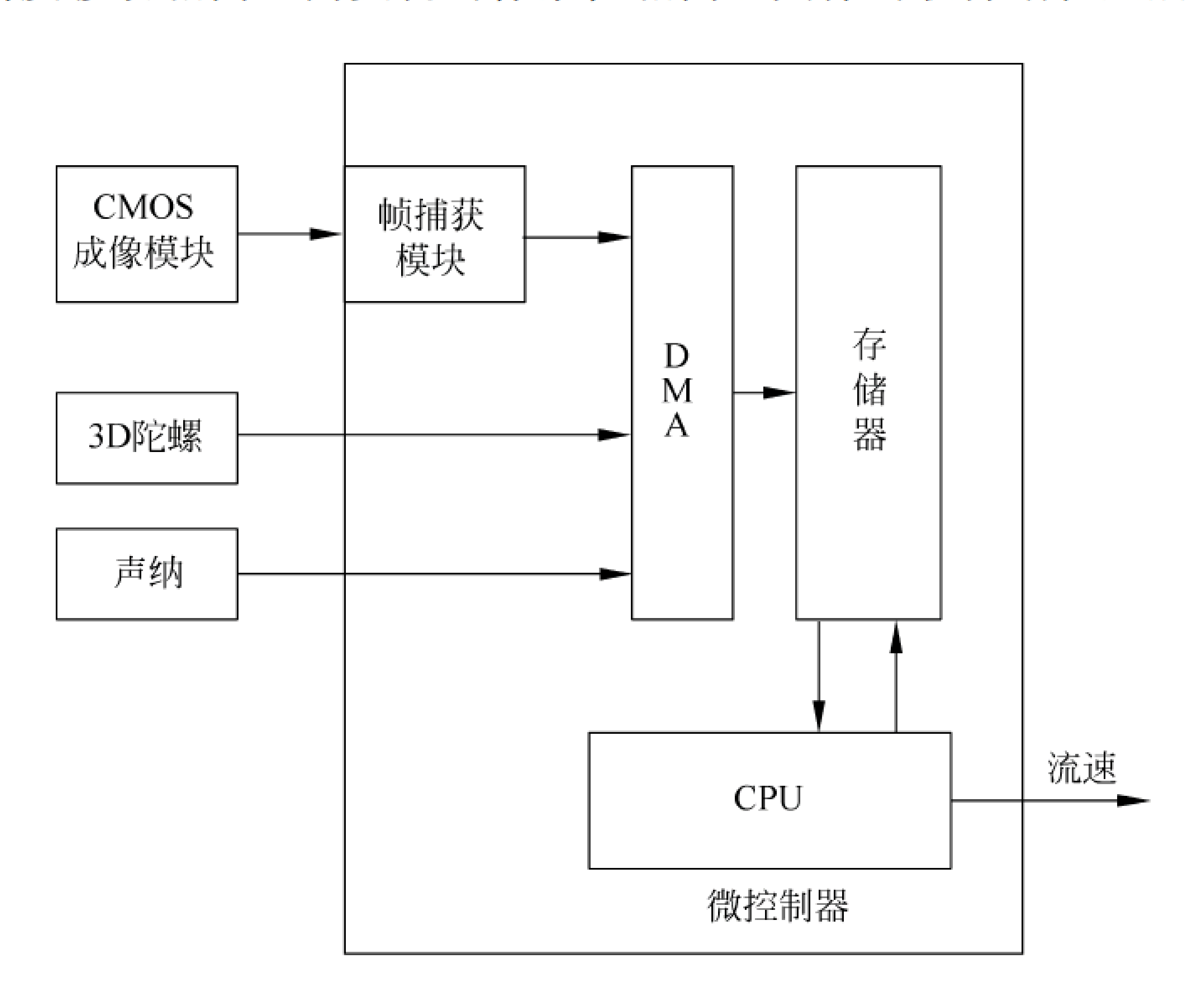


图 4.1.19　光流原理图

另一款为对地面要求相对来说更加接近真实环境的光流传感器——PX4Flow,串口传输数据。但需要对 MAVLink 协议进行解析,相对复杂一些,但已经进行过高度修正,具体的软件编程见第 5 章。

接下来介绍它们的结构和工作原理。

光流图像数据来自于 CMOS 机器视觉传感器。和 ARM Cortex M4 处理器的专用图像接口相连,以便微处理器能实时处理图像。帧捕获模块将抓取到的图像数据存在处理器存储器中,光流模块通过连续的两帧图像计算光流数据,通过细化获取的子像素和陀螺仪角速度对计算结果做进一步处理,最后将计算结果转化为以 m 为单位的信息。

图像数据通过并口送到微处理器,处理器帧捕获模块在像素时钟的控制下采取像素数据,并通过 DMA 送到存储器。DMA 设定了两个缓冲区,交替接收图像数据。存储器中只需存储当前和上次采集到的图像即可。

光流由连续的两帧图像数据进行计算。采用 SAD(Sum of Absolute Differences,绝对误差和)匹配算法。SAD 算法的基本流程如下。

① 构造一个小窗口,类似于卷积核。

② 用窗口覆盖左边的图像,选择窗口覆盖区域内的所有像素点。

③ 同样用窗口覆盖右边的图像,并选择覆盖区域的像素点。

④ 左边覆盖区域减去右边覆盖区域,并求出所有像素点差的绝对值的和。

⑤ 移动右边图像的窗口,重复③和④的动作(这里有个搜索范围,超过这个范围则跳出)。

⑥ 找到这个范围内 SAD 值最小的窗口,即找到了左边图像的最佳匹配的像素块。

通过对比当前图像和前一帧图像特定区域的 SAD 值,找出最接近的作为光流运算结果。为了支持不同的应用,对搜寻范围和模板大小可以进行配置。计算出匹配数据后,通过双线性插值(计算相邻两个像素点的平均值作为子像素)获取的子像素做进一步查询匹配,使精度达到亚像素级。陀螺仪角速度数据可以用来补偿纠正由于旋转引起的光流移动,使计算结果真实反映水平的移动。随着旋转方向的不同,SAD 查找窗口可以移动,这样使光流在水平方向查找范围很小的情况下允许的旋转速度更大。

SAD 算法将图像分成大小为 8×8 的特征模板。每次会搜寻正负 4 个像素的数据进行匹配运算。每个采样点会和 81 个四周的像素点进行 SAD 运算,并将值最小的作为匹配点。每帧处理 64 个采样点,对每个采样点采用直方图滤波,选出槽最大的值,值以像素为单位。在 1m 高的距离、帧率为 250、焦距为 16mm、搜寻范围为正负 4 个像素点的条件下,可测的最大地面速度为±1.5m。在基于像素点计算出最好的光流数据后,通过子

像素做进一步细分。子像素通过双线性插值获得,通过子像素在各个方向上的进一步匹配,结合基于像素计算的数据,一起作为最终的匹配结果。

光流所用镜头接口为 M12,焦距 16mm,FOV(视场角)为 21°,有红外过滤功能。摄像头采用的是镁光 MT9V034,像素点大小为 6μm,最大分辨率为 752H×480V。最大分辨率下的最高帧率为 60fps。有 binning 功能(是一种图像读出模式,将相邻像元感应的电荷加在一起,以一个像素的模式读出),在 4 倍 binning 下,分辨率为 188H×120V,最高帧率可到 250fps。binning 功能由摄像头自己完成,4 倍 binning 时,输出的图像每个像素点对应的为 4×4 个原始像素的均值。图像处理器采用 32 位的 STM32F407,处理器采用 Cortex M4F 核,168MHz 主频,192KB RAM,有浮点运算单元。陀螺仪选用 L3GD20,16 位的分辨率,最大测速到 2000°/s。此外,还有 EEPROM 用以存储光流参数。声纳传感器选用 Maxbotics 公司的 HRLV-EZ4,该传感器小巧、轻便。

ARM Cortex M4F 微处理器提供了一个可配置尺寸和像素深度的图像获取接口。采用 8 位每个像素的分辨率,以便于使用 ARM 一次可以处理 32 位的指令每次处理 4 个像素点。图像接口通过 DMA 将图像数据存放在处理器内部的存储空间中。Cortex M4 专门的整数向量指令在一个 CPU 时钟里一次可以并行处理 4 个像素的数据。

PX4Flow 是一款智能光学流动传感器(如图 4.1.20 所示),计算光学流的过程中采用了 4 倍分级和剪裁算法(4×4 分级图像算法),光流运算速度从 120Hz(室内)至 250Hz(室外);高感光度,有 24μm×24 μm 高像素。一个光流传感器所包含的部件如下。



图 4.1.20 PX4Flow 光流传感器

① STM32F405,168MHz Cortex M4F(单精度浮点,128 + 64KB RAM)。

② 752×480 MT9V034 机器视觉图像 CMOS 传感器,L3GD20 三轴陀螺。

③ 16mm M12 镜头(集成红外滤光片)。

④ 外形尺寸:45.5 mm×35mm。

⑤ 功耗:115mA/5V。

PX4Flow 模块使用 USB 和 UART 输出 MAVLink 协议包,波特率为 115 200。可以使用地面站 QGroundControl 从光流中读取数据。PX4Flow 光流传感器能通过检测视觉芯片感知区域的变化,测量图像的移动速度。通过测量暗点和光点经过芯片的速

度，就能够检测出图像移动的速度。它有强大的图像捕捉能力，也就意味着它能够处理更多样的地面情况，这使四旋翼飞行器在较为简单的地面达到稳定悬停提供了可能。

在与飞行控制板连接时，光流有 4 个引脚需要连接到飞行控制板上，分别是 5V 电源输入、GND、TX、RX。光流 TX 与飞行控制 UART-RX 相连，RX 连接 UART-TX。

如图 4.1.21 所示，通过地面站可读取到数值，确认飞机向正方向飞行，光流输出值为负值，所以在进行飞机悬停校正时，应该特别注意加上光流输出的位移量的正负。光流安装与飞行器正方向平行，摄像头对飞机尾部方向。

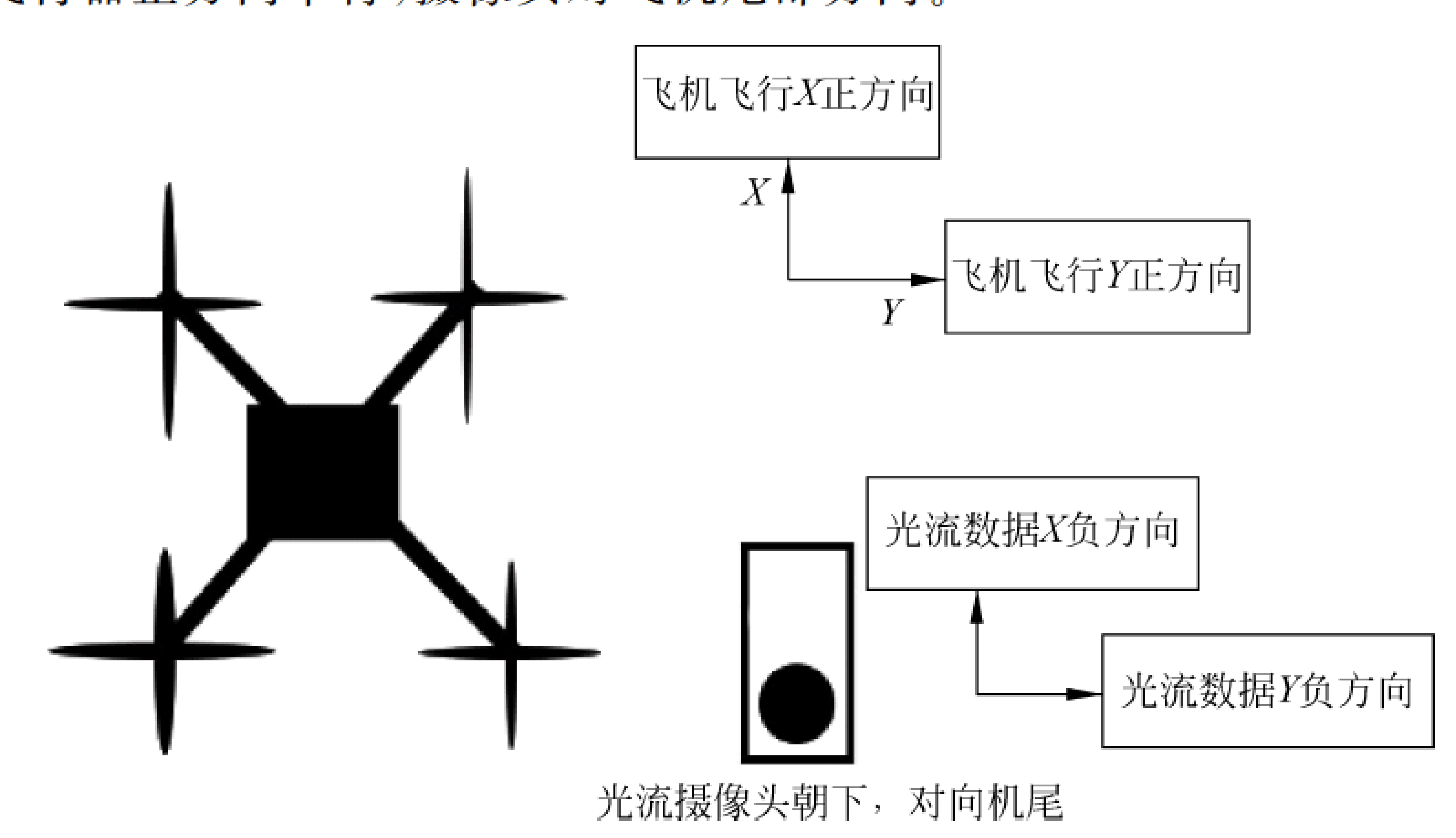


图 4.1.21　光流与四旋翼飞行器方向示意图

6. 视觉图像识别与跟踪模块

这里介绍的图像传感器模块是在 2016 年 TI 杯上海市大学生电子设计竞赛中用于图像识别而设计的，题目要求飞行器能识别出地面上的黑色三角形、圆形以及大小两个正方形，所以图像识别部分是基于这一功能来阐述的。当然读者在理解了本小节内容之后也可以根据 OV2640 摄像头和 STM32F407ZGT6 芯片的特点，以本节内容为基础编写代码实现自己需要的其他图像的识别功能。

图像传感器模块由 ARM 芯片 STM32F407ZGT6、SRAM 芯片 IS62WV51216、OV2640 摄像头、惯性导航模块 JY-901、电源模块、UART 串行接口、LED 指示灯等组成，系统框图如图 4.1.22 所示。图像传感器模块是基于体积小、重量轻、功耗低的要求而设计的，所以电路必须最简化，而开发调试的工作在带 LCD 显示屏的地面站上完成。

1）STM32F407ZGT6 简介

STM32F407ZGT6 是图像传感器模块的核心芯片，选用这款芯片主要是因为它有 8～14 位 54MB/s 摄像头 DCMI 接口，图像数据可直接由 DMA 采集，CPU 主频有

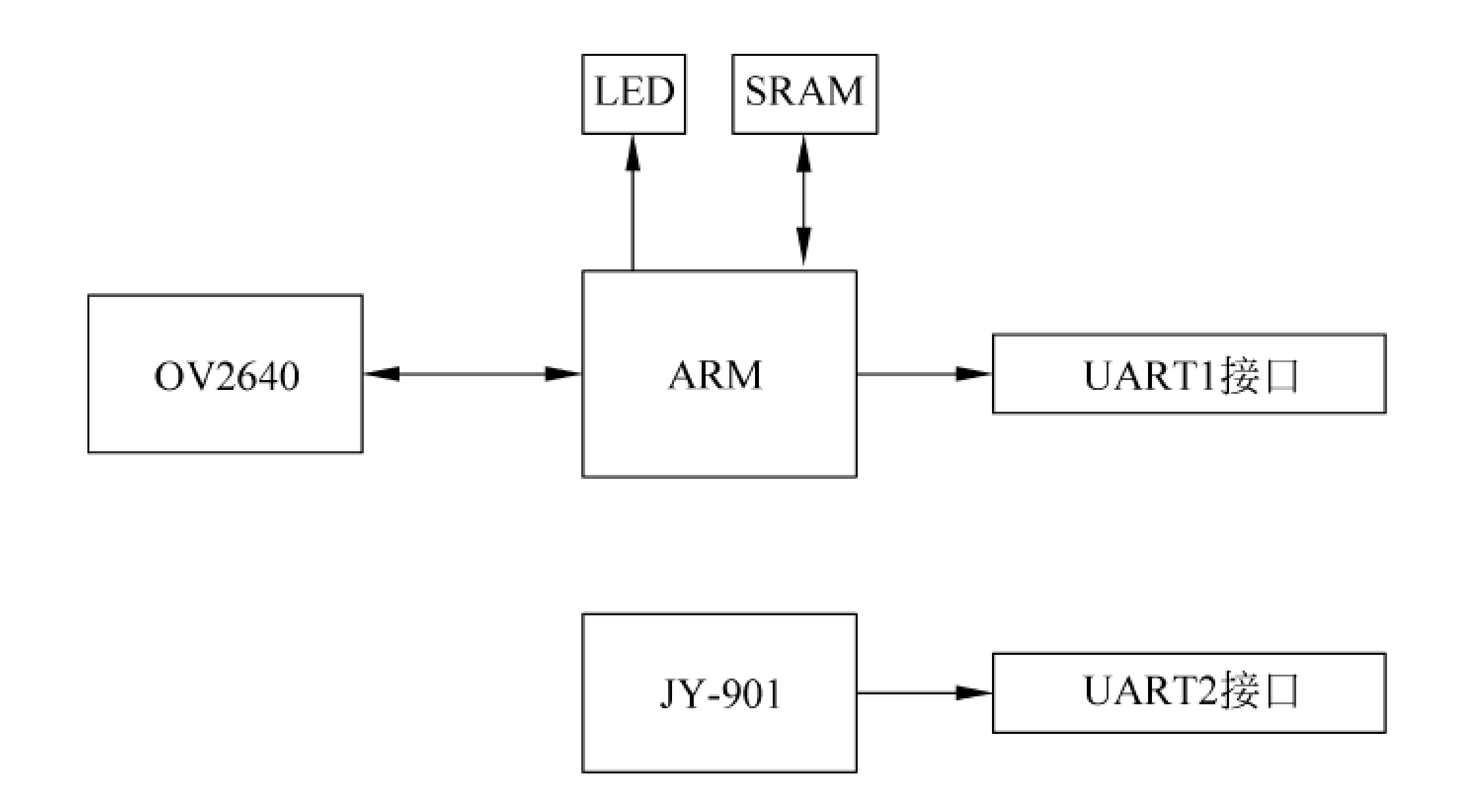


图 4.1.22　图像传感器模块框图

168MHz，对特定几何图像识别的处理速度足够了。另外，该芯片集成 FPU 和 DSP 指令，并具有 192KB SRAM、1024KB Flash、12 个 16 位定时器、2 个 32 位定时器、2 个 DMA 控制器(共 16 个通道)、3 个 SPI、2 个全双工 I^2S、3 个 I^2C、6 个串口、2 个 USB(支持主/从)、2 个 CAN、3 个 12 位 ADC、2 个 12 位 DAC、1 个 RTC(带日历功能)、1 个 SDIO 接口、1 个 FSMC 接口、1 个 10/100M 以太网 MAC 控制器、1 个摄像头接口、1 个硬件随机数生成器以及 112 个通用 IO 口等。欲知详细情况请查阅 STM32F407ZGT6 数据手册。

2) OV2640 简介

OV2640 摄像头模块采用 1/4 英寸的 200 万像素 CMOS 传感器制作，具有高灵敏度、高灵活性、体积小、工作电压 3.3V、支持 JPEG/RGB565 格式输出等特点。通过 SCCB 总线控制，模块尺寸 27mm×27mm。OV2640 实物如图 4.1.23 所示，OV2640 尺寸如图 4.1.24 所示，OV2640 原理图如图 4.1.25 所示。



图 4.1.23　OV2640 实物

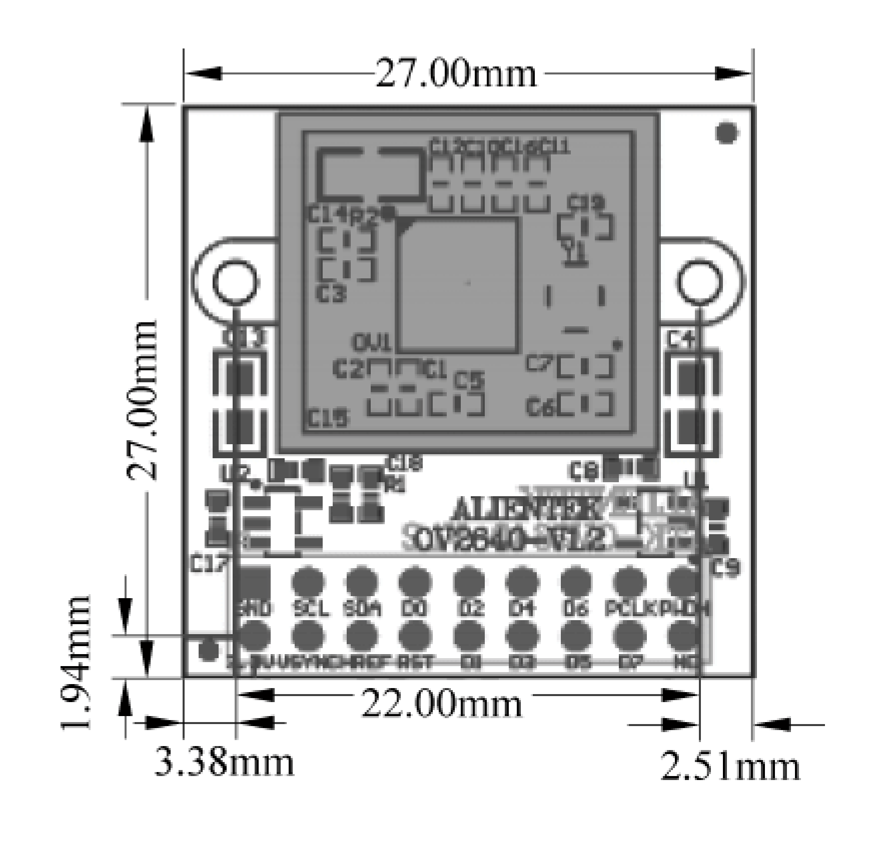


图 4.1.24　OV2640 尺寸

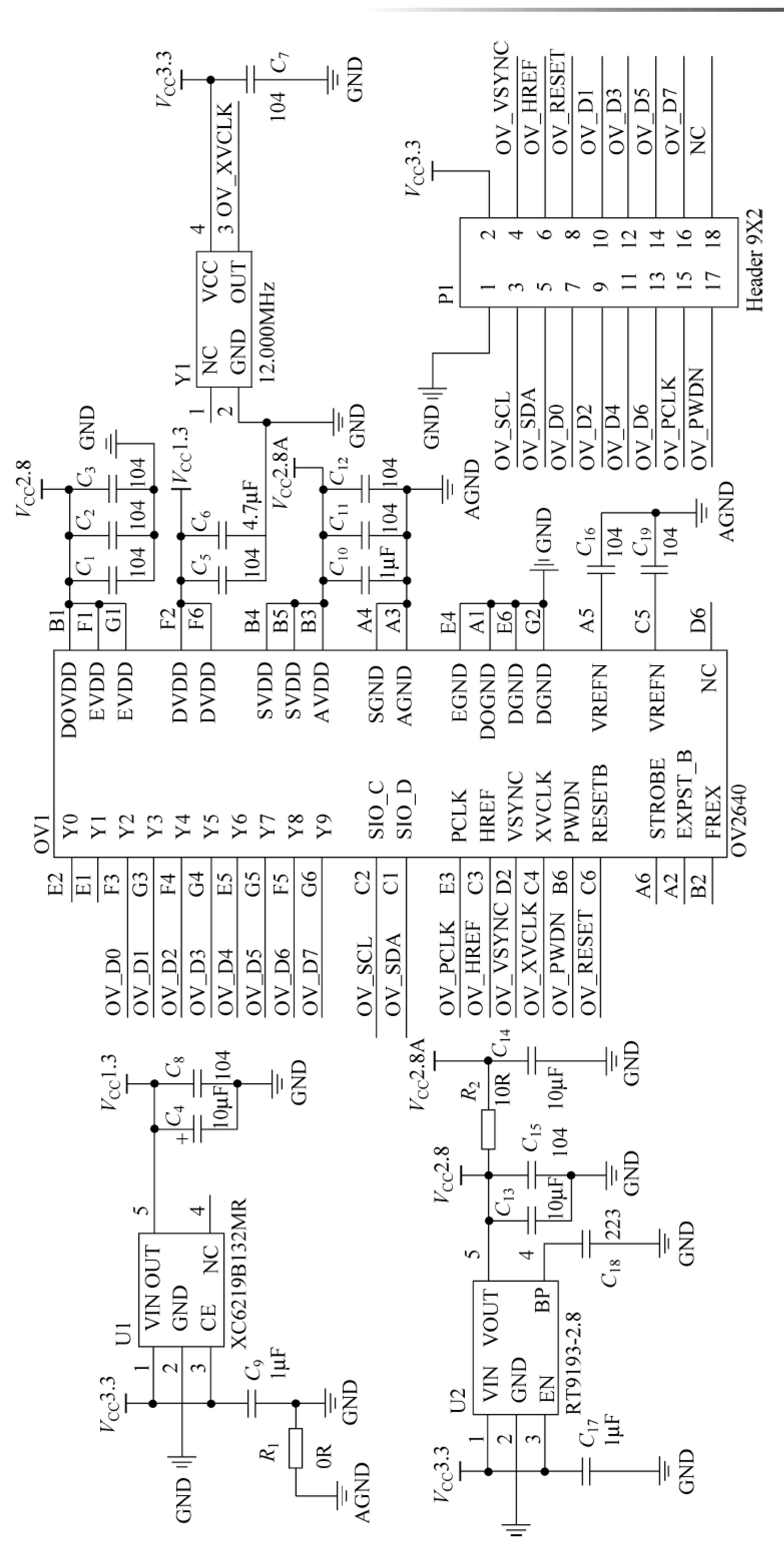


图 4.1.25　OV2640 原理图

OV2640 的 UXGA 图像最高达到 15f/s(SVGA 可达 30f/s,CIF 可达 60f/s)。用户可以完全控制图像质量、数据格式和传输方式。所有图像处理功能过程包括 γ 曲线、白平衡、对比度、色度等都可以通过 SCCB 接口编程。

OV2640 的特点如下。

① 高灵敏度、低电压适合嵌入式应用。

② 标准的 SCCB 接口,兼容 I^2C 接口。

③ 支持 RawRGB、RGB(RGB565/RGB555)、GRB422、YUV(422/420)和 YCbCr(422)输出格式。

④ 支持 UXGA、SXGA、SVGA 以及按比例缩小到从 SXGA 到 40×30 的任何尺寸。

⑤ 支持自动曝光控制、自动增益控制、自动白平衡、自动消除灯光条纹、自动黑电平校准等自动控制功能。同时支持色饱和度、色相、γ 曲线、锐度等设置。

⑥ 支持闪光灯。

⑦ 支持图像缩放、平移和窗口设置。

⑧ 支持图像压缩,即可输出 JPEG 图像数据。

⑨ 自带嵌入式微处理器。

3) 图像传感器模块硬件简介

四旋翼飞行器图像传感器模块基于 STM32F407ZGT6+OV2640 处理器设计,5V 输入经内部 3.3V 稳压芯片后为模块供电,尺寸长、宽、高为 78mm×59mm×35mm。OV2640 用于采集图像,以 SCCB 接口方式连接 STM32F407ZGT6 处理器,图像经 DMA 采集后存入 SRAM 芯片 IS62WV51216,图像经处理、识别后将判别结果信息通过串口传输到飞行控制板,从而决定飞行器的飞行动作。气压计模块 JY-901 是自带处理器的户外飞行高度测量仪,为了安装方便,将它集成在图像传感器电路板上,通过串口与飞行控制板相连。图像传感器模块实物如图 4.1.26 所示。

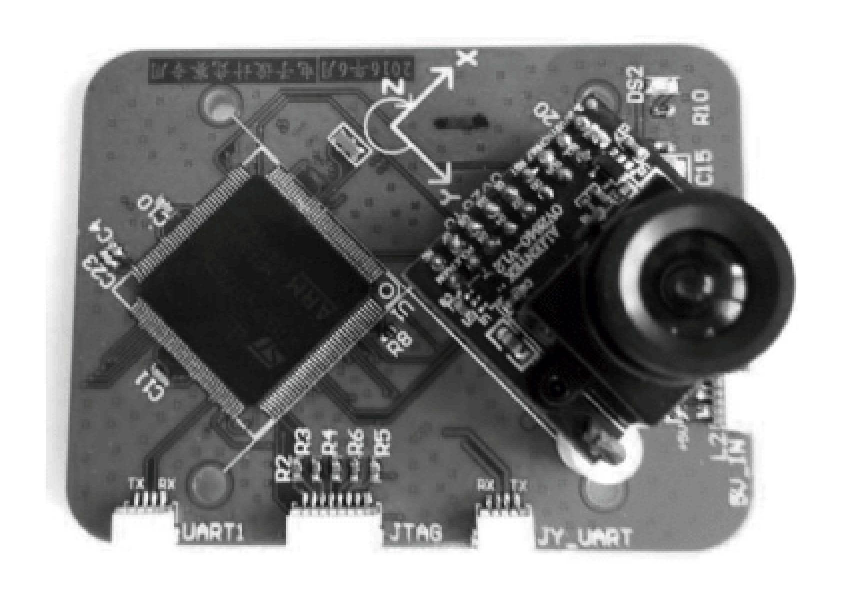


图 4.1.26 图像传感器模块实物

4) 图像识别程序分析

(1) 图像采集

图像采集流程框图如图 4.1.27 所示。因为处理器 CPU 的大量时间耗费在图像处理和识别上,所以将图像的采集工作交给 DMA(直接内存访问)完成,DMA 是以提供外

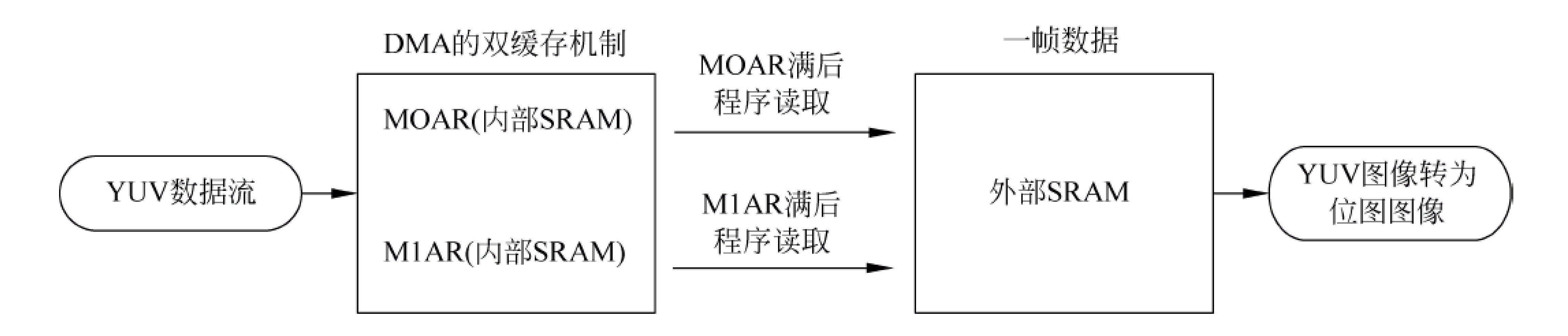


图 4.1.27　图像采集流程框图

设和内存、内存和内存之间的高速数据传输的，数据可以在没有任何 CPU 干预下通过 DMA 进行传输。DMA 控制器基于一个复杂的总线矩阵架构，结合了功能强大的双 AHB 主总线架构与独立的 FIFO，一个负责内存访问，另一个负责外设访问。

一幅图像如果用内存储存，那么内存是不够用的，必须将图像数据保存在外部 SRAM，从 DMA 直接存外部 SRAM 较慢，图像数据会部分丢失，必须使用 DMA 的双缓冲机制来读取。DMA 接收来自 OV2640 的 YUV 数据流，首先使用 M0AR（内存 1）来存储，当 M0AR 满了以后产生中断，自动切换到 M1AR（内存 2），同时程序读取 M0AR（内存 1）的数据转存到外部 SRAM；当 M1AR 满了以后，又切回 M0AR，同时程序读取 M1AR（内存 2）的数据到外部 SRAM；依次循环，直到产生帧中断，结束一帧数据的采集，并读取全部剩余数据到外部 SRAM，完成一次 YUV 数据的采集。由于采用了双缓冲机制，M0AR 和 M1AR 数组可以定义得比较小，一幅图像可以让 M0AR 和 M1AR 依次循环存储多次直至接收完成。最后，将存储在外部 SRAM 的 YUV 数据，经过二值化处理转为位图图像（YUV 图像一个像素为两个字节，而位图图像一个像素为 1/8 个字节）。该过程的主要程序流程框图如图 4.1.28 所示。

（2）图形识别

在图像采集函数中用到了 Figure_Identify（bwimage，borderx，bordery）函数，该函数对需要识别的圆形、三角形、大小正方形先进行目标框定，为防止死循环做了删除断点区域处理。通过对框定区域内图形面积和周长的计算来判定图形的形状，并提取该图形的中心坐标。求识别图形的面积与周长后判别是什么形状图形的计算公式为

$$\mathrm{Rad} = \frac{4\pi A}{L^2} \tag{4.1.1}$$

式中，L^2 为周长的平方，是为了消除高度对判别的影响；4π 是为了使 Rad 值归一化。根据 Rad 值的范围可判别是什么形状图形。

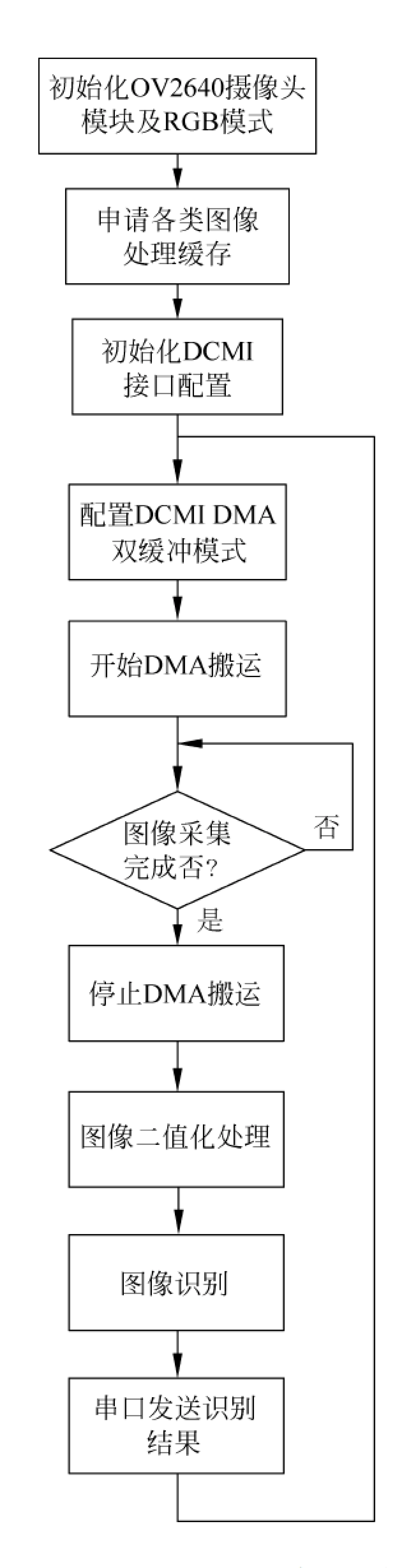


图 4.1.28 图形处理程序流程框图

① Rad>0.86 则判定为圆形。

② 0.65<Rad≤0.86 则判定为正方形。

③ Rad≤0.65 则判定为三角形。

此函数图形识别程序设计流程框图如图 4.1.29 所示。

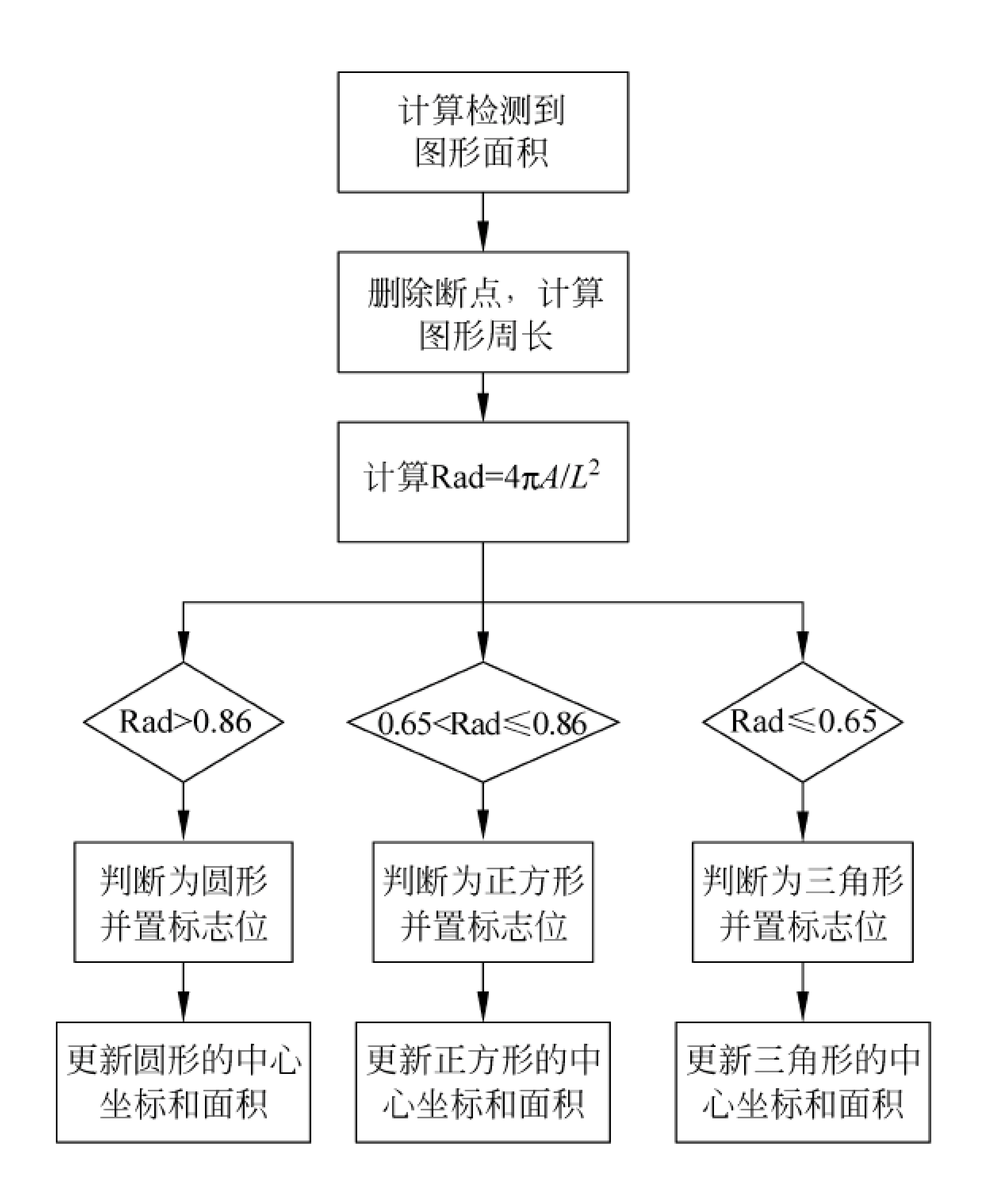


图 4.1.29　图形识别程序设计流程框图

一个图形被采集后，处理生成一个二值化图像矩阵（其中，图形部分全为白，背景部分为黑，为了便于分析，将图形部分每个点都用 1 表示，背景点用 0 表示）。在对待识别图形目标框定后，面积的计算只要数“1”的个数即可，周长的计算需要搜索出图形的轮廓，就是要把二值矩阵里的“1”的路径找到。采取从左到右、从上到下的扫描顺序搜索轮廓的第一个点。按此种扫描方式，本书按逆时针的方向（也可用顺时针，算法类似）搜索轮廓。设轮廓第一点在此 3×3 模板的 0 位置，那么第二点只可能从 1 位置开始搜索，而不可能出现在 5、6、7、8 位置。那么在剩下的 1、2、3、4 位置上肯定能找到第二个轮廓点。至此，轮廓的第一点和第二点找到。本书便可以提出沿外法线方向搜索下一个轮廓点进而搜索出整个图形轮廓。用外法线方法搜索整个轮廓的好处是：可以沿着图形的边沿把每一个“1”值点都快速找到而不用去搜索 X_0 八邻点的每一点。还避免了搜索出的图形轮廓变形，如按指定位置搜索轮廓点会把图形内部的“1”值点误认为轮廓点。

算法如图 4.1.30 和图 4.1.31 所示，实现如下。

① 按从上到下、从左到右的顺序扫描二值矩阵，得到第一个轮廓点，将它的行列坐标存入轮廓数组，数组指针加 1。

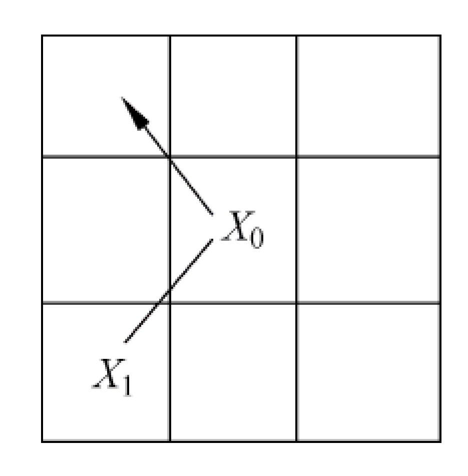


图 4.1.30　法线方向

7	6	5
8	0	4
1	2	3

图 4.1.31　模板各点编号

② 令第一点在此 3×3 模板的 0 位置，在第二点可能的 1、2、3、4 位置上扫描，找到第二个轮廓点，将它的行列坐标存入轮廓数组，数组指针加 1。

③ 根据前两个搜索到的轮廓点，设这两点为 X_0 和 X_1，X_0 始终置于 3×3 模板的中心 0 位置。搜索下一个轮廓点，对应到模板的编号，如果 X_1 在 1 位置，那么下一个轮廓点的搜索应从 7 位置开始，在此模板中心的八邻点中，沿逆时针方向搜索出值为"1"的图形点，简记为(0，1，7)、(0，2，8)、(0，3，1)、(0，4，2)、(0，5，3)、(0，6，4)、(0，7，5)、(0，8，6)，括号里的前两个数字表示 X_0 与 X_1 的位置，后一个数字表示外法线方向的搜索开始位置。把搜索到的这个轮廓点的行列坐标存入数组，数组指针加 1。如果找到的这个点与第一个轮廓点的行列坐标相同，即表示搜索回到了起点，整个轮廓已搜索出，该算法结束；否则，继续转入③。

计算周长＝偶数链码×1＋奇数链码×1.414。在轮廓搜索的过程中，根据轮廓已经将链码分为奇偶链码，两点相距为 1 个单位时为偶链码，两点相距为$\sqrt{2}$个单位时为奇链码，用上公式便可求出周长。

7. 航拍视频传输模块

具有视觉功能的四旋翼飞行器有很多用途，这里讨论一种简单的飞行器采集和发送视频图像的方法，并用手机显示图像，系统总体框架图如图 4.1.32 所示。

为了图像能实时显示，视频图像采集和传输使用 WR703N 无线路由器，在 OpenWrt 的嵌入式 Linux 系统下进行程序开发。用舵机云台使得摄像头可以分别在两个自由度上进行 180°的转动。

1）硬件构成与选择

要想实现无人机航拍并将视频实时传输这一功能，摄像头与传输方式的选择至关重要。考虑到要将航拍功能应用于四旋翼飞行器上，而飞行器的有效载荷有限，因此应选用体积较小的、空间利用率高的 WiFi 发射装置。

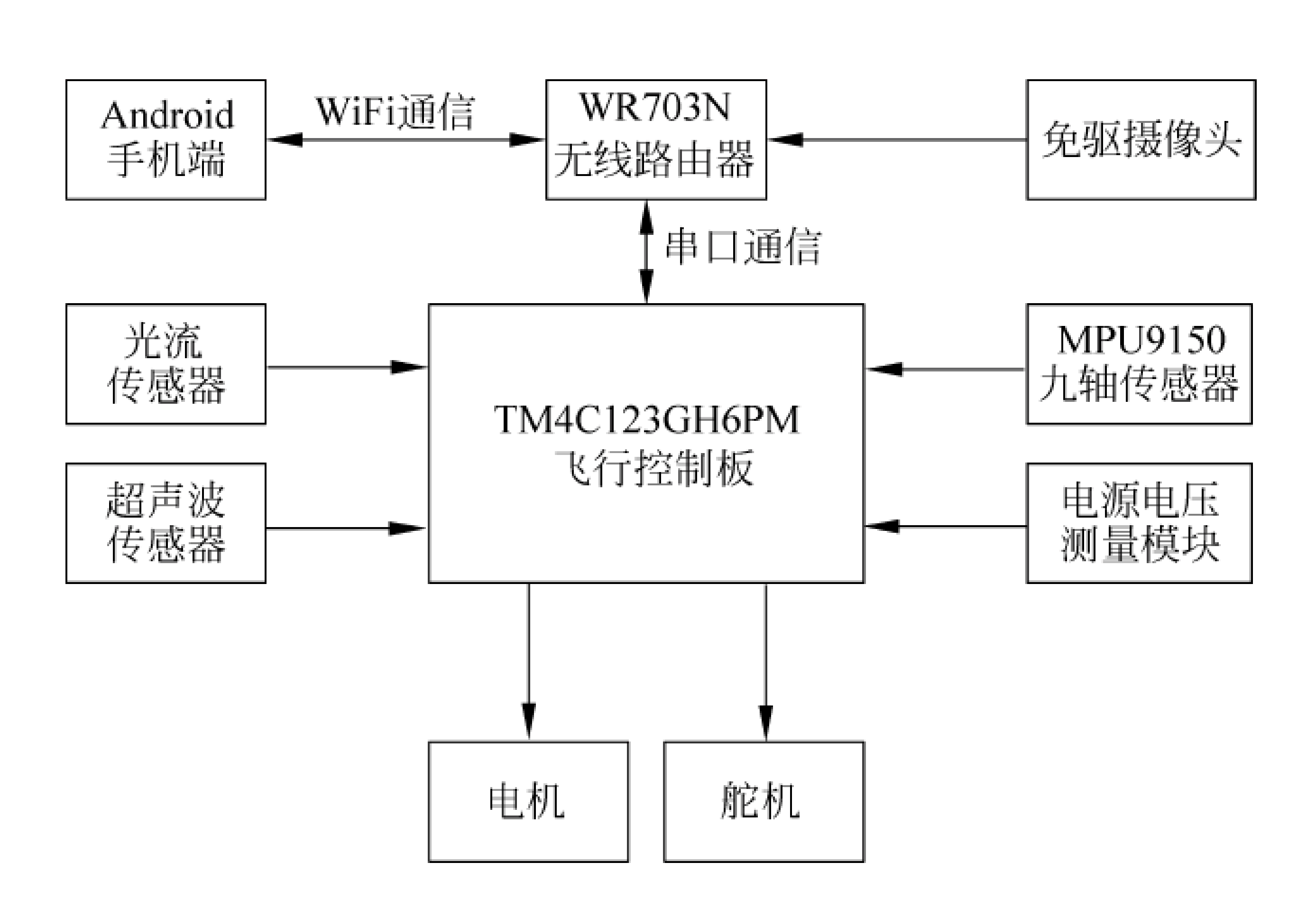


图 4.1.32　系统总体框架图

经调查对比，本设计采用 WR703N 无线路由器作为 WiFi 信号发射器。它不仅体积小、集成度高、利于安装固定，而且在软件部分采用了开源的 OpenWrt，程序编写灵活简单。

同样可以刷写 OpenWrt 固件的无线路由器还有 WR720N，以下从各方面对两者进行比较。

① 两者均有 32MB 的 RAM 和 4MB 的闪存。

② TL-WR720N 有两个网线接口，可以通过有线外接台式机，而且有内置电池，可以给一些移动终端供电，但不能给自身供电。不过可以用外置电池供电使用。TL-WR703N 只有一个网线接口。

③ 二者均支持 3G 功能，并且最高传输速率都是 150Mb/s。

④ 网络接口。两者 LAN/WAN 复用接口的速率相同，都可以达到 10/100Mb/s，但 WR703N 只有一个复用接口。

⑤ WR703N 个头比 WR720N 小巧得多，质量也很轻。

⑥ WR720N 有时等待识别需要很长时间。

综合上述各种因素考虑，本设计最终选择 WR703N 无线路由器作为 WiFi 信号发生装置，实物如图 4.1.33 所示。其中，电源由电调供电，USB 接口连接摄像头，3 根导线接出来的接头连接飞行控制

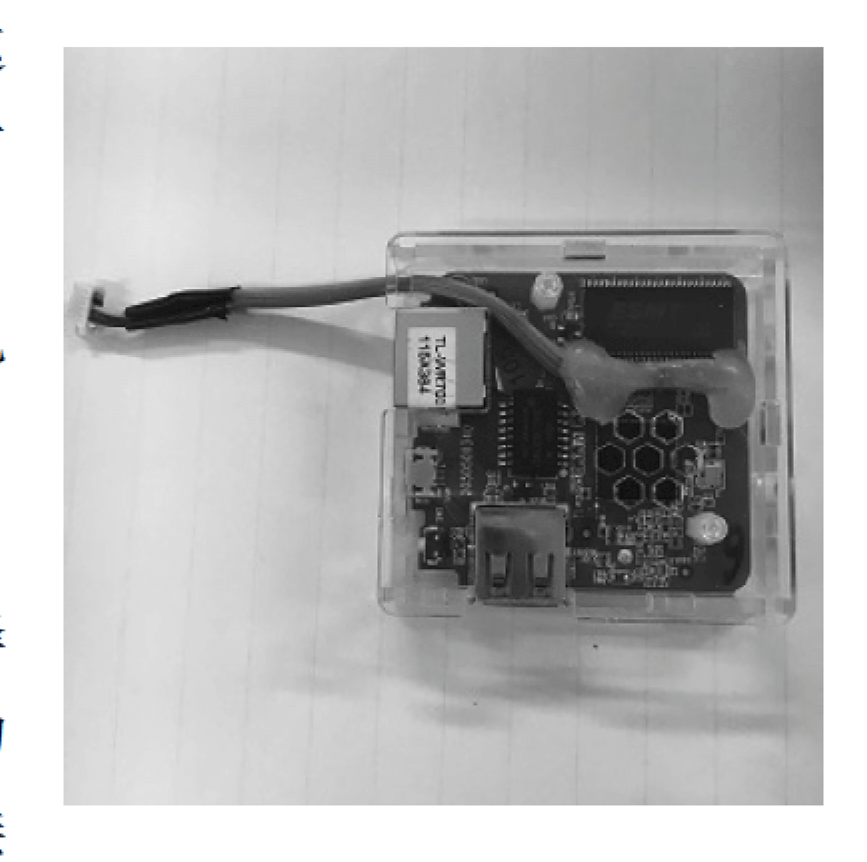


图 4.1.33　WR703N 路由器

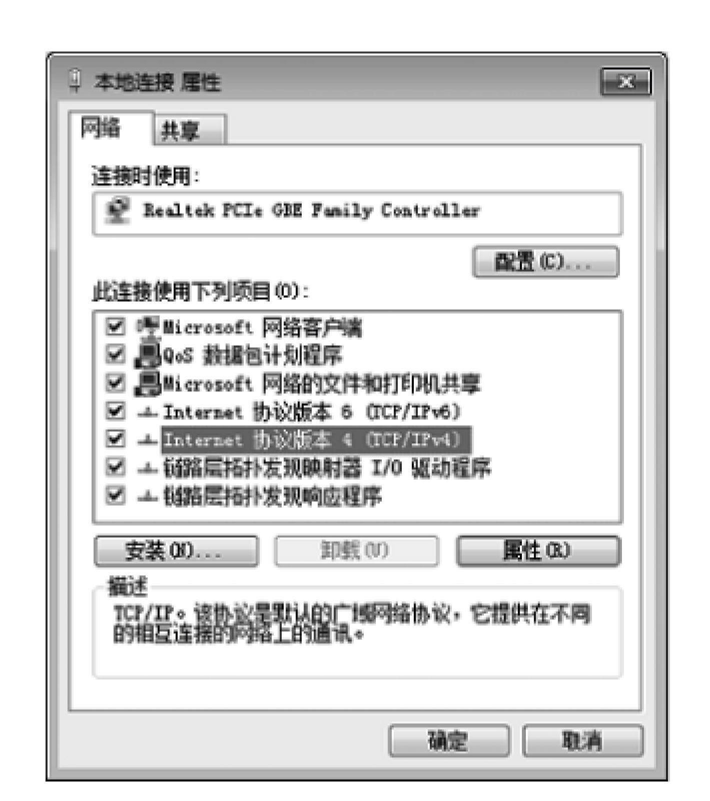
板，进行串口通信。

2）OpenWrt 的使用

（1）OpenWrt 固件刷写

刚买来的 WR703N 无线路由器并不能直接使用，要对其进行 OpenWrt 固件的刷写。

① 找到“本地连接 属性”对话框，对“Internet 协议版本 4（TCP/IPv4）属性”进行正确的配置，如图 4.1.34 所示。



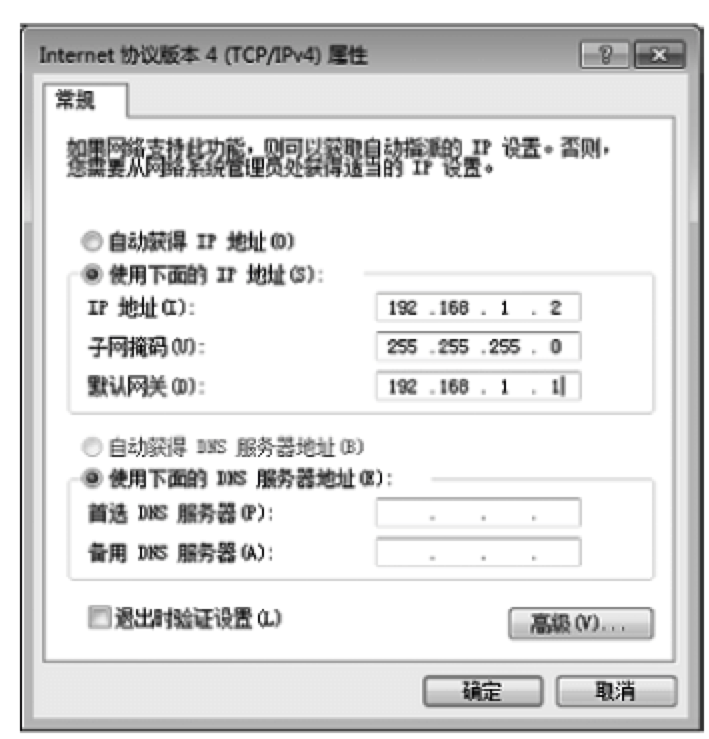


图 4.1.34 “本地连接 属性”界面（左）和“Internet 协议属性”界面（右）

② 将烧写的固件放入 Tftp32 文件夹内，如图 4.1.35 所示。Tftp32 相当于一个多功能的网络服务器包，主要用于不同服务器间的资源传输。



图 4.1.35 放置固件到目的路径

③ 在计算机与路由器之间用网线和 TTL 线连接，登录 Putty，如图 4.1.36 所示。Putty 是一款在 Windows 平台下访问 Linux 系统的软件。

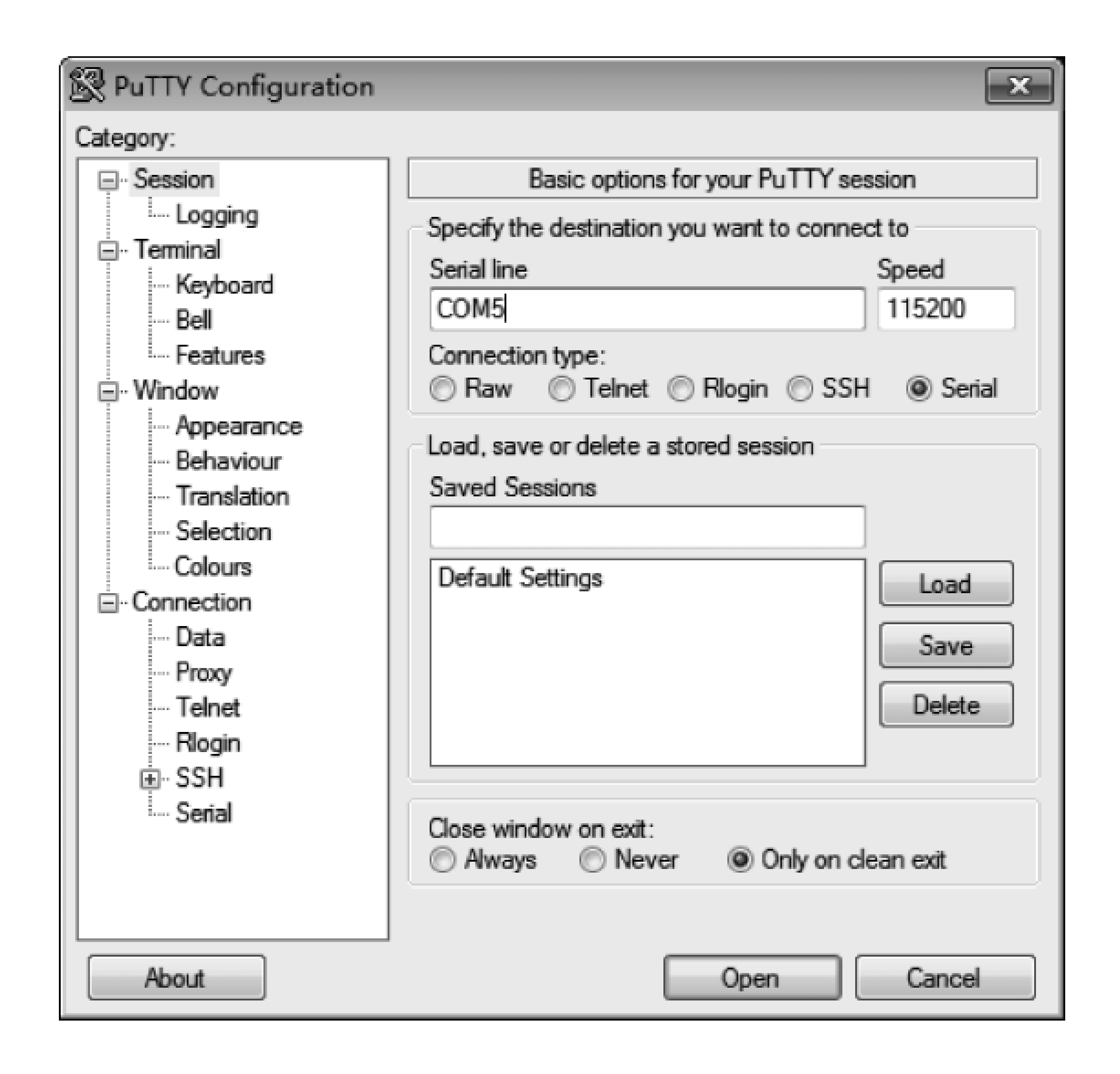


图 4.1.36　使用 Putty 登录

④ 对系统进行适当的配置即可完成固件刷写，OpenWrt 的启动界面如图 4.1.37 所示。软件 SercureCRT 功能与 Putty 相似。

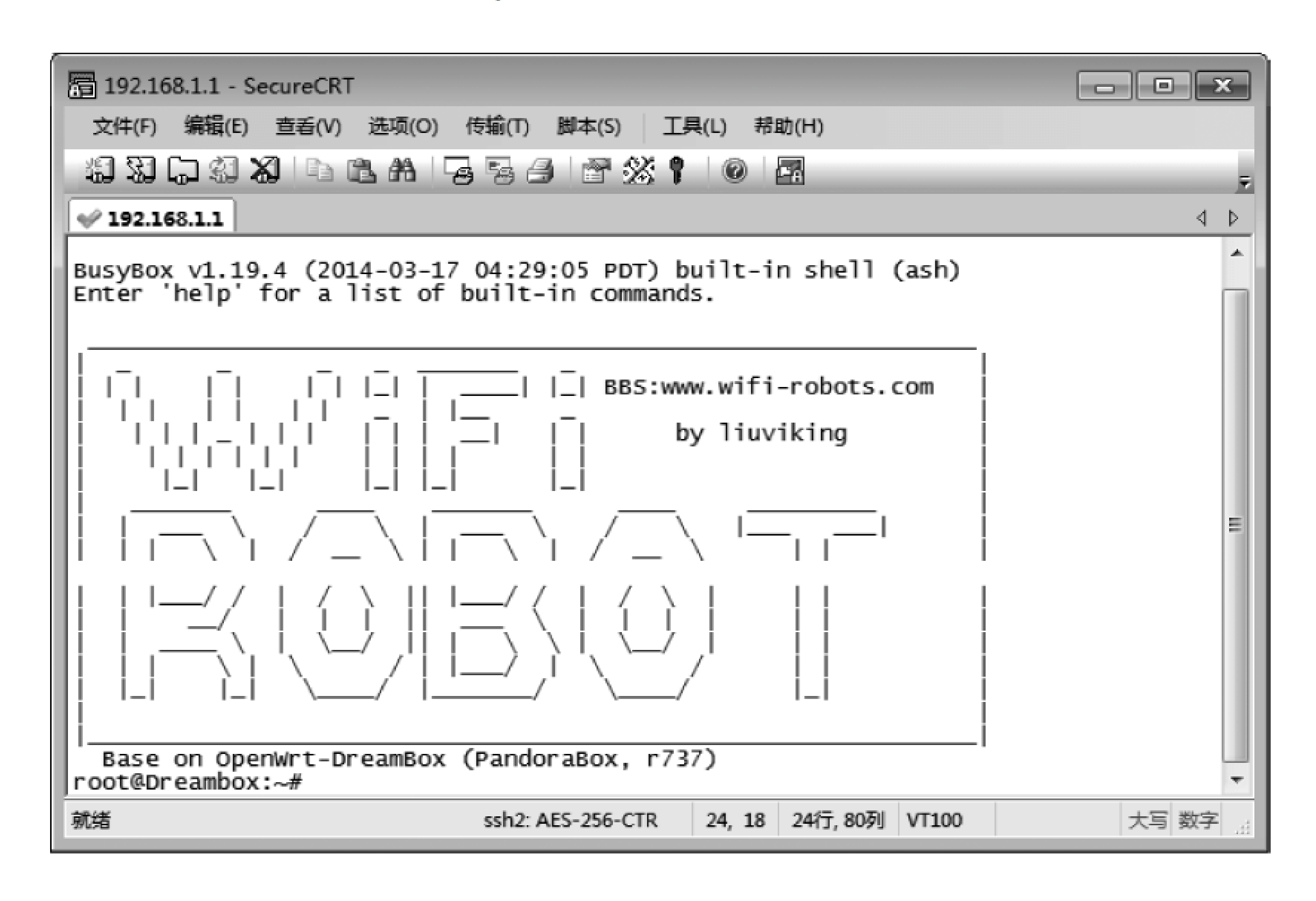


图 4.1.37　OpenWrt 启动界面

OpenWrt是一个集成化的Linux系统，在功能上有很好的扩展性，可进行挂在U盘、串口透传、智能家居、中继转发等应用开发。它包含了上千个功能包，也提供了很多已经编译好的软件供使用。与普通发行版路由器不同的是，OpenWrt是一个从无到有、配置灵活、使用简便、功能多样的完全开源的Linux发行版。

（2）备份安装bin文件

① 用Putty进入路由以后，输入cat /proc/mtd查看分区状况。里面说明了各个分区的内容，如图4.1.38所示。

```
root@Dreambox:~# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00020000 00010000 "u-boot"
mtd1: 000e1bd4 00010000 "kernel"
mtd2: 002ee42c 00010000 "rootfs"
mtd3: 000b0000 00010000 "rootfs_data"
mtd4: 00010000 00010000 "art"
mtd5: 003d0000 00010000 "firmware"
```

图4.1.38 分区情况

至于firmware分区，则是包含了所有的kernel和rootfs以及rootfs_data和art，如表4.1.5所示。

表4.1.5 分区情况说明

分区名称	描　述
u-boot	设备初始化程序+引导程序代码本身
kernel	内核，顾名思义是一个系统的最核心部分，存放的都是对内核进行管理的核心信息，这些信息都是以二进制形式存放的
rootfs	完整的系统文件包含只读和可写
rootfs_data	在rootfs中的可写部分的位置，为配置文件所在区，rootfs_data相当于/overlay
art	EEPROM分区，在Atheros方案中这个分区保存了无线的硬件参数
firmware	完整的固件位置包含了除u-boot和art之外全部的内容

② 输入以下几行命令就可以备份整个.bin格式的back_firmware文件了，如图4.1.39所示。

```
root@Dreambox:~# cd /tmp
root@Dreambox:/tmp# dd if = /dev/mtd5 of = /tmp/back_firmware.bin
7808 + 0 records in
7808 + 0 records out
```

```
root@Dreambox:/tmp# ls
TZ               hosts      resolv.conf       sysinfo
back_firmware.bin lock      resolv.conf.auto
dhcp.leases      log        run
etc              overlay    state
```

图4.1.39 文件备份目录

③ 通过WinSCP传到Windows系统，如图4.1.40所示。WinSCP是不同系统之间文件传输处理的工具。

back_firmware.bin	2014/3/17 星期...	BIN 文件	3,904 KB

图 4.1.40　备份文件显示

(3) 编译一个 Hello World 程序

① 在 Ubuntu 下，新建名为 hello_world.c 文件，编写相应的程序：

```
vi hello_world.c
```

② 输入以下内容：

```
#include <stdio.h>
int main(char argc, char * argv[])
{
    int i = 1;
    while(1){
        printf("Hello world!!! %d\n",i);
        if(i<10)
            i++;
        else
            i=1;
        sleep(1);
    }
    return 0;
}
```

③ 用 mips 指令编译"hello_world.c"文件，并生成"hello_world"可执行文件 mip-openwrt-linux -gcc hello_world.c -o hello_world。

④ 用 WinSCP 将文件传输到/tmp 目录下。

⑤ 在开发板的串口终端下执行"hello_world"文件：

```
/tmp/hello_world
```

(4) 交叉编译并生成 IPK 文件

在嵌入式开发过程中学会使用交叉编译的方法是十分必要的。交叉编译通常用于两个有不同控制语言的平台之间，先在 A 台机器上对程序进行编译，生成可执行文件，再将可执行文件转移到 B 机器上来运行。起初要从官网上下载交叉编译工具链并进行安装，工具链通常包括编译器、连接器、解释器和调试器 4 个部分。

① 切换到 openwrt 根目录，执行下列命令：

```
cd./package/utils            //进入 package/utils 目录
mkdir hello_world            //创建一个名为"hello_world"的文件夹,用于放置安装包源码
cd hello_world               //进入相应文件夹
mkdir src                    //新建一个名为"src"的文件夹
vi src/hello_world.c         //在 src 目录下新建一个名为 hello_world.c 文件,内容同上
```

② 在 src 目录下新建一个 Makefile，输入相应的内容：

```
vi src/Makefile
```

③ 在当前目录下新建一个 Makefile 文件并进行编写。Makefile 如同脚本文件，其中定义编写了一系列系统需要执行的操作命令。

④ 返回至 openwrt 文件夹，执行命令：

```
make menuconfig
```

选择已经加进去的安装包：

```
utilities - - - <
<M> hello_world ..................... Hello world - prints a hello world message
```

保存并退出。

⑤ make V=s。等待编译完成后，可以在 openwrt 目录下的 base 文件夹内找到生成的安装包“/hello_world_1.0_ar71xx.ipk”。

⑥ 将 ipk 包传输与安装到开发板上即可运行。

更改权限 root@Dreambox:/# chmod u+x /tmp/hello_world 。

运行结果如图 4.1.41 所示。

```
root@Dreambox:/#  /tmp/hello_world
Hello world!!!1
Hello world!!!2
Hello world!!!3
Hello world!!!4
Hello world!!!5
Hello world!!!6
```

图 4.1.41 Hello World 运行结果

(5) OpenWrt 高级功能

除了基本的 Hello World 程序的运行，OpenWrt 这个嵌入式 Linux 系统还有很多高级功能可以拓展，下面介绍其中 3 种模式。

① AP 模式。接入点模式，可以不用设置 SSID 和密码，关闭 DHCP 服务器，外设通过 WiFi 连接开发板。

② Router。路由器模式，需输入 SSID 和密码才可以进入，开启 DHCP 服务器，外设通过 WiFi 连接开发板。

③ 二级无线路由器模式设置。开启 DHCP 服务器，连接开发板的设备和上级路由器处于不同网络，一般不能互通。

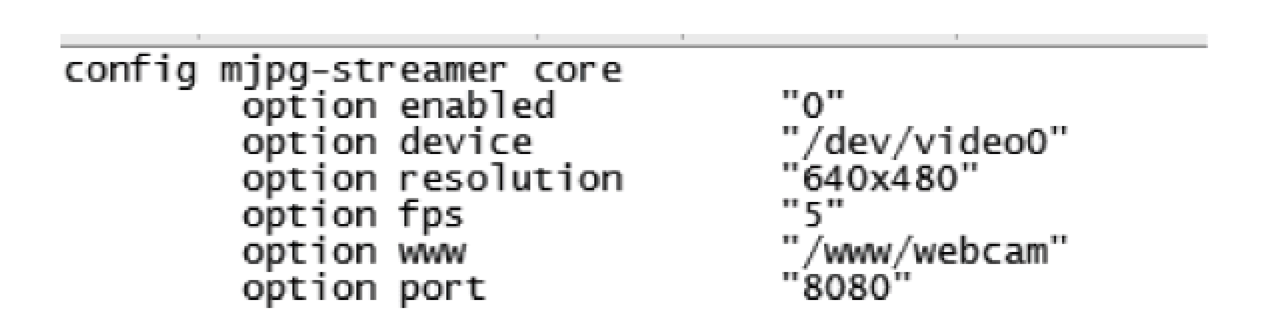
3）OpenWrt 挂载摄像头

要实现视频传输必须要安装相关功能的 IPK，如表 4.1.6 所示。

表 4.1.6　视频传输相关 IPK 介绍

IPK 名称	描　　述
kmod-video-core	摄像头内核模块，UVC 驱动依赖包
kmod-video-videobuf2	UVC 驱动依赖包
kmod-video-uvc	UVC 驱动
libpthread	mjpeg-streamer 依赖包
libjpeg	mjpeg-streamer 依赖包
mjpg-streamer	mjpeg-streamer 功能安装包

插入摄像头，根据系统打印信息判断开发板是否支持该型号的摄像头，必须选择 UVC 摄像头。在安装完摄像头之后会出现“/etc/config/mjpg-streamer”配置文件，如图 4.1.42 所示。



```
config mjpg-streamer core
        option enabled          "0"
        option device           "/dev/video0"
        option resolution       "640x480"
        option fps              "5"
        option www              "/www/webcam"
        option port             "8080"
```

图 4.1.42　mjpg-streamer 文件内容

文件中的内容分别配置了设备名称、摄像头的分辨率、帧率、访问目录与访问端口。在连接上路由器发射的 WiFi 信号之后，用浏览器打开网址 http://192.168.1.1:8080/?action=stream，即可看到实时传输的视频。

4.1.3　遥控器介绍

当前最常用的遥控器是 2.4GHz 频段的遥控器，常见的有天地飞、富斯等品牌，遥控

器握持在使用者的手中，无线或通过信号线连接至地面站，经地面站处理将 PPM 信号并解析成真正的遥控数据，实现遥控器对四旋翼飞行器的控制。而接收机安装在四旋翼飞行器机身上以接收发射机的信号，并且将各个通道的数据用 PWM 波的形式输送到四旋翼飞行器对应的接收引脚。

若使用遥控器与对应接收机进行飞行操控，需要注意以下细节。

① 使用前需要进行对码。

② 由于 2.4 GHz 是全球免费频段，因此为了防止多个遥控器之间的互相干扰，在使用前通常需要进行发射机和接收机的对码操作。具体来说，则是接通接收机和发射机电源之后进行的一系列操作，以便使接收机和发射机之间存在信号的验证，避免干扰。接收机通常接到 5V 的电源上。

③ 发射机通道及反向设置。

④ 当前的遥控器一般至少是 6 通道。而通道数量的多少和误码率等也是选择遥控器的标准。四旋翼飞行器至少需要 4 个通道，剩下的通道主要用于控制相机、云台以及其他设备。

Futaba T6j 遥控器包括一台 6 通道的 Futaba 6j 发射机和一个 2.4GHz 的 R2006GS 高压接收机，发射机和接收机通过内置序列号相匹配，一台发射机只能对一个特定的接收机发射信号。一般地，控制 4 轴的 4 个通道是在发射机的摇杆上面，而其他的通道往往以旋钮或者拨动开关的形式放在遥控器面板的上方。

根据摇杆与通道的配置关系，发射机可以分为“左手油门(美国手)”与“右手油门(日本手)”两种，如图 4.1.43 所示，左边为日本手，右边为美国手。

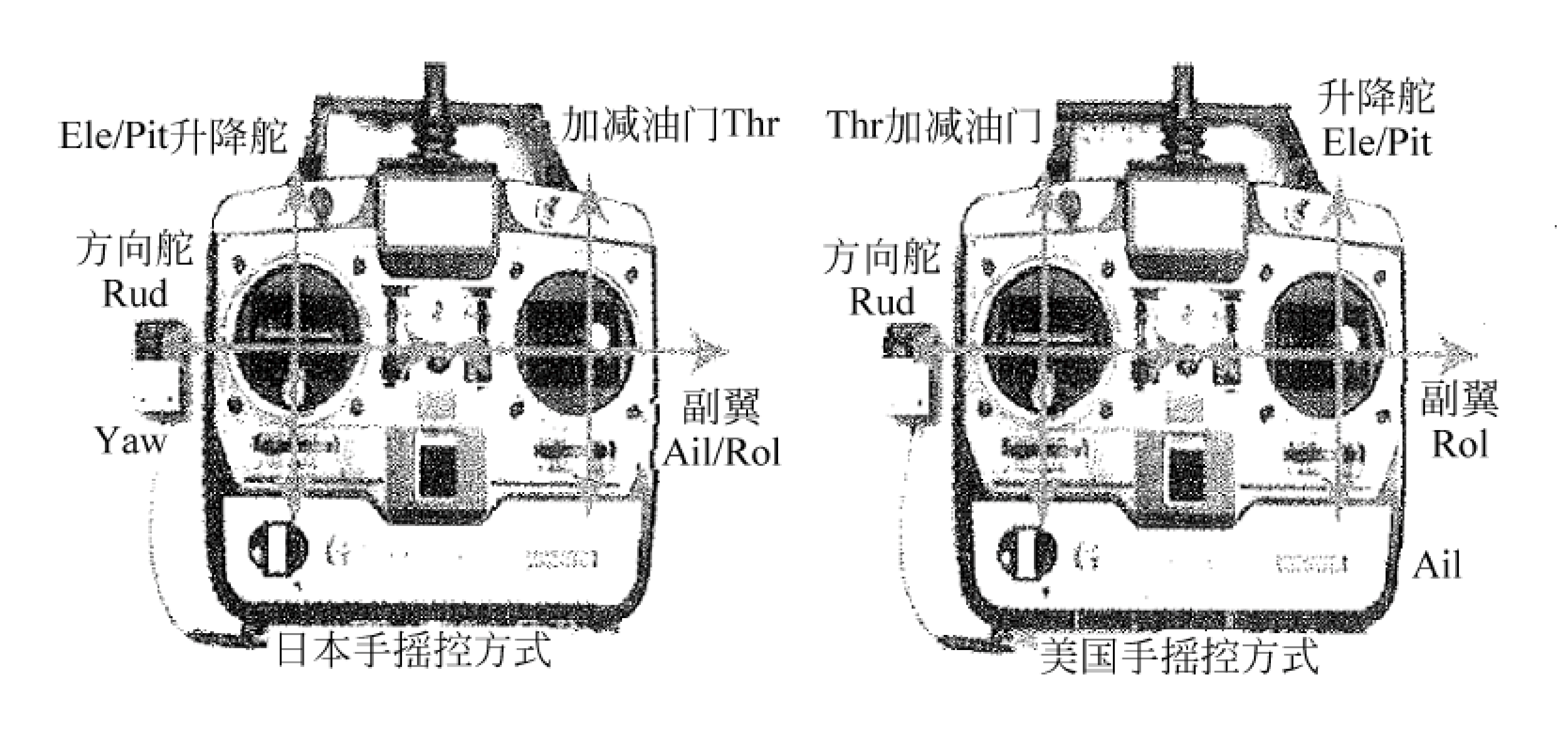


图 4.1.43 常用 2.4GHz 航模遥控器各通道示意图

一般认为，日本手由于两手分开控制 Rol/Pit 这两个最重要的姿态量，比较适合航拍和新手使用，而美国手更适合固定翼。

Futaba 6j 发射机通道及功能如下。

① 油门通道、舵向通道（偏航）、副翼通道（翻滚）和升降通道（俯仰），这 4 个通道用来控制四旋翼飞行器的飞行姿态。其中油门通道和舵向通道集成在一起，副翼通道和升降通道集成在一起，分别由一个两自由度调节杆控制。每个通道对应一个微调旋钮，通过微调各通道，可以使遥控器工作在最佳状态。

② 第五通道为一个两段式开关，有高、低两种状态，第六通道为一个旋钮式开关，由低到高可调。通过调节这两个开关，结合 GUI 设置，可实现四旋翼飞行器如定高、定点、自动返航等不同飞行模式。R2006GS 高压板接收机是 1.4GHz 的无线接收器，其有两根天线，两根天线末端尽量保持垂直关系，以便更好地接收无线信号。

4.2　机架的构造与电机的选用

4.2.1　机架结构与设备安装

机架是安装前文中所述各个部分的重要载体，某种程度上也将影响四旋翼飞行器的性能。较大四旋翼飞行器的机架一般采用金属或者玻璃纤维和工程塑料制作，而微型四旋翼飞行器的机架一般采用 PCB 直接充当。

较大四旋翼飞行器的机架一般可以自己动手做，主要分为中心板、四臂、电机固定座以及起落架等部分，如图 4.2.1 所示。

中心板是机架的核心，用于固定四旋翼飞行器的四臂。同时，一般上面会有间隔 45mm×45mm 的安装孔用于安装飞行控制板。一般情况下，中心板有两层，而四臂通过中心板上的固定螺栓夹到两层中心板的中间。四臂的末端，一般有电机座固定螺孔和起落架固定螺孔。这里用的通常是直径为 3mm 的长螺杆螺钉。安装时需要注意以下一些细节。

① 拧螺钉要紧一点，维持较好的机架机械强度；因为四旋翼飞行器的四臂外形会影响其力学特性。

② 飞行控制板安装的时候要放平，有些飞行控制系统在起飞前检查水平位置，并且飞行控制系统应该尽量放置在四旋翼飞行器机架的中心位置。

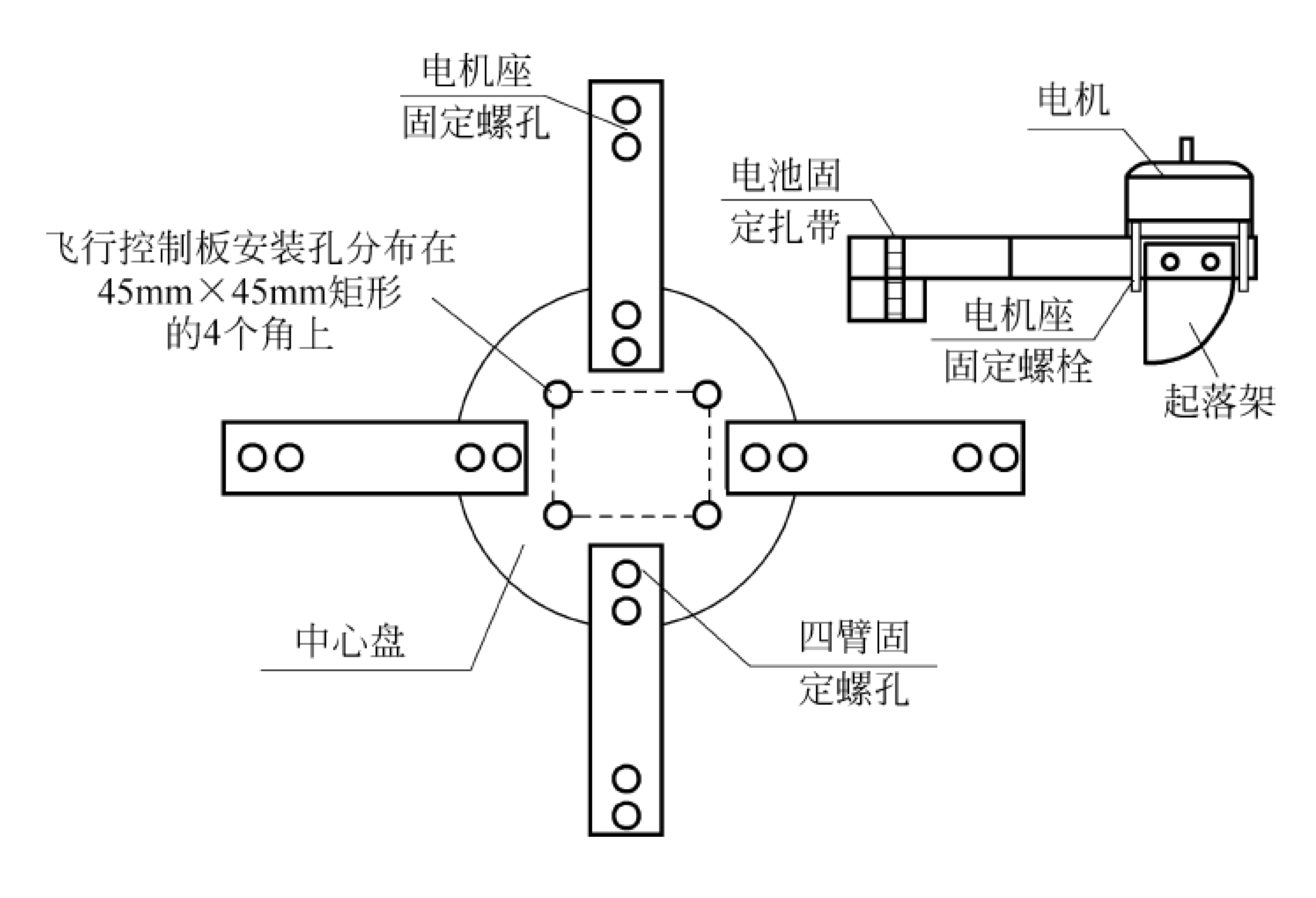


图 4.2.1　机架结构

③ 电池要绑稳定，也应尽量靠近中间位置，以便保持较好的重心结构。

由于对重量的要求较高，微型四旋翼飞行器的机架通常可以直接用自身的 PCB 充当，或者是用碳纤维、工程塑料等比较轻且比较坚固的材料加上比较小的 PCB 构成。使用 PCB 作为机架时，可以选择比较薄的 PCB，如 0.8mm 厚度，这样能降低重量。同时，PCB 本身可以设计成机架的样子，将中心板、四臂、电机安装座都包含在内。

4.2.2　电机与桨的选用

四旋翼飞行器的控制，关键在于姿态控制，控制电机转速是最后的一步。电机作为四旋翼飞行器控制的最终执行部件，对四旋翼飞行器姿态的调节起到直接的作用。选择无刷电机的时候，有以下一些数据是需要注意的。图 4.2.2 所示为新西达出品的一款无刷电机。



图 4.2.2　新西达 2212 无刷电机

1. 尺寸

无刷电机在型号命名上就反映了其具体大小。例如，常用的新西达 2212 电机，含义就是，直径为 22mm，转子的高度为 12mm。一般而言，越大的电机其转速和扭力也就越大。

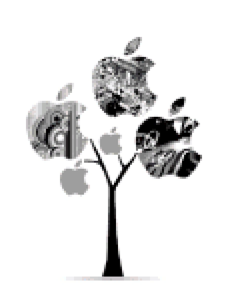
2. KV 值

无刷电机 KV 值定义为“转速/V”，意思为当输入电压增加 1V 时，无刷电机空转转速增加的转速值。例如，KV1000 的无刷电机，代表电压为 11V 的时候，电机的空转转速为 11 000r/min。KV 值越大，速度越快，但扭力越小；KV 值越小，速度越慢，但扭力越大。450～600mm 轴距，2kg 左右的四旋翼飞行器，一般采用新西达 2212-KV1000 电机搭配 1045 螺旋桨就可以了。

电机右侧为固定座，用于将电机固定到机架上面；子弹头用于安装螺旋桨；电机上的 3 根线，即右边示意图与电调连接的电机驱动线。如果调换其中任意两根的位置，则电机的转动方向将会发生变化。转动方向与正反桨的转动方向及搭配非常重要，因此安装时一定要注意。安装完毕后，4 个螺旋桨吹起的风应该都是向下的。

电机与螺旋桨的搭配是非常复杂的问题，螺旋桨越大，升力越大，但需要更大的扭力来驱动；螺旋桨转速越高，升力越大；电机的 KV 值越小，扭力越大。因此大螺旋桨需要低 KV 值的电机，小螺旋架需要高 KV 值的电机。如果高值的电机带大桨，扭力不足，那么电机和电调就很容易烧掉；如果低 KV 值的电机带小桨，电机电调没大问题，但产生的升力严重不足，可能无法起飞。因此，需要通过大量试验才能得出电机与螺旋桨的搭配关系。仍然以新西达 2212-1000KV 电机为例，通常可搭配 1045 桨使用。1045 的含义是，螺旋桨的直径是 10 英寸(1in＝2.54cm)，螺距是 4.5 英寸。也就是说，螺旋桨每旋转一圈前进 4.5 英寸。实际使用中还可以使用三叶螺旋桨等。

第 5 章



四旋翼飞行器软件系统设计

5.1 开发工具简介

5.1.1 CCS

CCS(Code Composer Studio)是 TI 公司研发的一款具有环境配置、源文件编辑、程序调试、跟踪和分析等功能的集成开发环境,能够帮助用户在一个软件环境下完成编辑、编译、链接、调试和数据分析等工作。当前的最新版本为 CCSv6,它具有以下特点。

① Windows 下工作,类似于 Visual C++的集成开发环境。

② 采用图形接口界面,有编辑工具和工程管理工具。

③ 它将汇编器、链接器、C/C++编译器、建库工具等集成在一个统一的开发平台中。

④ CCS 所集成的代码调试工具具有各种调试功能,能对 TMS320 系列 DSP 进行指令级的仿真和可视化的实时数据分析。

⑤ 丰富的输入输出库函数和信号处理库函数。

⑥ CCS C5000 是专门用来开发 C5000 系列 DSP 系统(C54x 和 C55x)的平台。

5.1.2 IAR

IAR 的 Embedded Workbench 系列是一种增强型一体化嵌入式集成开发环境,其中完全集成了开发嵌入式系统所需要的文件编辑、项目管理、编译、链接和调试工具。IAR

公司独具特色的 C-SPY 调试器，不仅可以在系统开发初期进行无目标硬件的纯软件仿真，也可以结合 IAR 公司推出的 J-Link 硬件仿真器，实现用户系统的实时在线仿真调试。

IAR 的 Embedded Workbench 系列适用于开发基于 8 位、16 位以及 32 位微处理器的嵌入式系统，其集成开发环境具有统一界面，为用户提供了一个易学易用的开发平台。IAR 公司提出了"不同架构，唯一解决方案"的理念，用户可以针对多种不同的目标处理器，在相同的集成开发环境中进行基于不同 CPU 的嵌入式系统应用程序开发，有效提高工作效率，节省工作时间。IAR 的 Embedded Workbench 系列还是一种可扩展的模块化环境，允许用户采用自己喜欢的编辑器和源代码控制系统，链接定位器(XLINK)可以输出多种格式的目标文件，使用户可以采用第三方软件进行仿真调试和芯片编程。

5.1.3　Keil

Keil 公司是一家业界领先的微控制器(MCU)软件开发工具的独立供应商。Keil 公司由两家私人公司联合运营，分别是德国慕尼黑的 Keil Elektronik GmbH 和美国德克萨斯的 Keil Software Inc。Keil 公司制造和销售种类广泛的开发工具，包括 ANSI C 编译器、宏汇编程序、调试器、连接器、库管理器、固件和实时操作系统核心(Real-time Kernel)。有超过 10 万名微控制器开发人员在使用这种得到业界认可的解决方案。其 Keil C51 编译器自 1988 年引入市场以来成为事实上的行业标准，并支持超过 500 种 8051 变种。

Keil 公司 2005 年由 ARM 公司收购。而后 ARM Keil 推出基于 μVision 界面，用于调试 ARM7、ARM9、Cortex-M 内核的 MDK-ARM 开发，作为控制领域的开发工具。

2009 年 2 月发布 Keil μVision4，它引入灵活的窗口管理系统，使开发人员能够使用多台监视器，并提供了视觉上的表面对窗口位置的完全控制的任何地方。新的用户界面可以更好地利用屏幕空间和更有效地组织多个窗口，提供一个整洁、高效的环境来开发应用程序。新版本支持更多最新的 ARM 芯片，还添加了一些其他新功能。

5.2 飞行控制板控制系统软件总体设计

5.2.1 总体框架

由于飞行控制板控制软件对于实时性的要求非常高,采用 C 语言进行系统软件的设计,在主程序中执行各类硬件和软件的初始化函数,初始化完成后进入 while(1)循环。除了数据通信和超声波测距以外,其余飞行控制板所要执行的功能均在 400Hz 的主时间中断中依次进行,数据通信模块则会触发端口中断,超声波模块则会在数据采集到之后触发中断处理数据并存入变量。采用一个主时间中断,免去了对于中断优先级复杂的设置,重要的处理程序不会被其他中断所打断,易于确认每一次中断是否在下一次中断到来前执行完毕,方便了系统的计时,也确保了飞行控制系统的稳定性。

5.2.2 初始化

1. 电调的初始化

在新的电调安装完毕后,需要根据电调手册所提供的操作要求,对电调进行初始化。在程序中,设置一个电调初始化的函数以供调用,在主函数执行完成芯片基本初始化和 PWM 输出初始化之后调用该函数,即可完成对电调的初始化。在电调初始化的过程中,为了防止因操作失误导致人员受伤,通常不带桨进行初始化,初始化完成后,隐去电调初始化函数。

电调初始化就是设定电调的最高转速和最低转速,使 4 个电调实现对 PWM 波解析的同步,使四旋翼飞机能够实现平衡和稳定。

2. 各类软、硬件的初始化

要使一个飞行控制系统正常开始工作,需要对软件和硬件作一些初始化。软件方面,要初始化各类参数(如 PID 参数、油门限值等)、中断及其优先级等。硬件方面,不仅要对芯片本身的时钟、PWM 输出、UART 传输等进行初始化,还要针对外部对应的硬件,初始化相应用于数据交换的端口,对于某些硬件,还要通过数据端口,初始化这些硬件内部的参数,如 nRF 的设置。其流程框图如图 5.2.1 所示。

5.2.3　中断处理

1. 主时间中断

主时间中断依次调用各类函数以实现各种功能。对于不需要以400Hz的频率调用的硬件或不能以太高频率使用的硬件来说，可以设置一个计数器定时处理。

例如，if(count0%10==1)的语句，在每若干个主中断中仅执行一次，以此来达到降低频率的效果。对于不必要以如此高频率执行的而执行一次又占用大量时间的多个函数，可以用这种方法使其分别在不同的时钟周期中进行，防止中断运行时间溢出，影响姿态解算和调整的稳定性。

主中断流程框图如图5.2.2所示。

2. 超声波中断

超声波模块在主中断触发超声波模块后会发出一束超声波，根据超声波模块接收到经过地面反射的超声波，超声波模块会向微处理器发送一段脉冲，脉宽正比于超声波发送和接收的时间差，当微处理器接收到这一脉冲后，进行一次输入输出中断，处理脉冲波数据获得高度值，并且存入变量，等待主中断中的定高控制函数进行调用。

3. nRF 中断

nRF在接收到数据时，会向微处理器发送信号，此时微处理器产生一个端口中断，判断nRF正常连接并给nRF正常连接的标志位置1，在主中断中接收nRF的数据的函数中，判断到nRF正常连接则会读取nRF中缓存的数据，该标志位也用于自动锁定飞机(当nRF没有连接时飞机自动锁定)。

4. UART 中断

在设定的UART数据端口接收到数据时，会产生中断，将数据存入缓存区，并置标志位，主中断中检测到标志位不为0，则会读取缓存区中的数据。

5.2.4　飞行控制程序中的重要变量列表

飞行控制程序中的重要参数如表5.2.1所示。

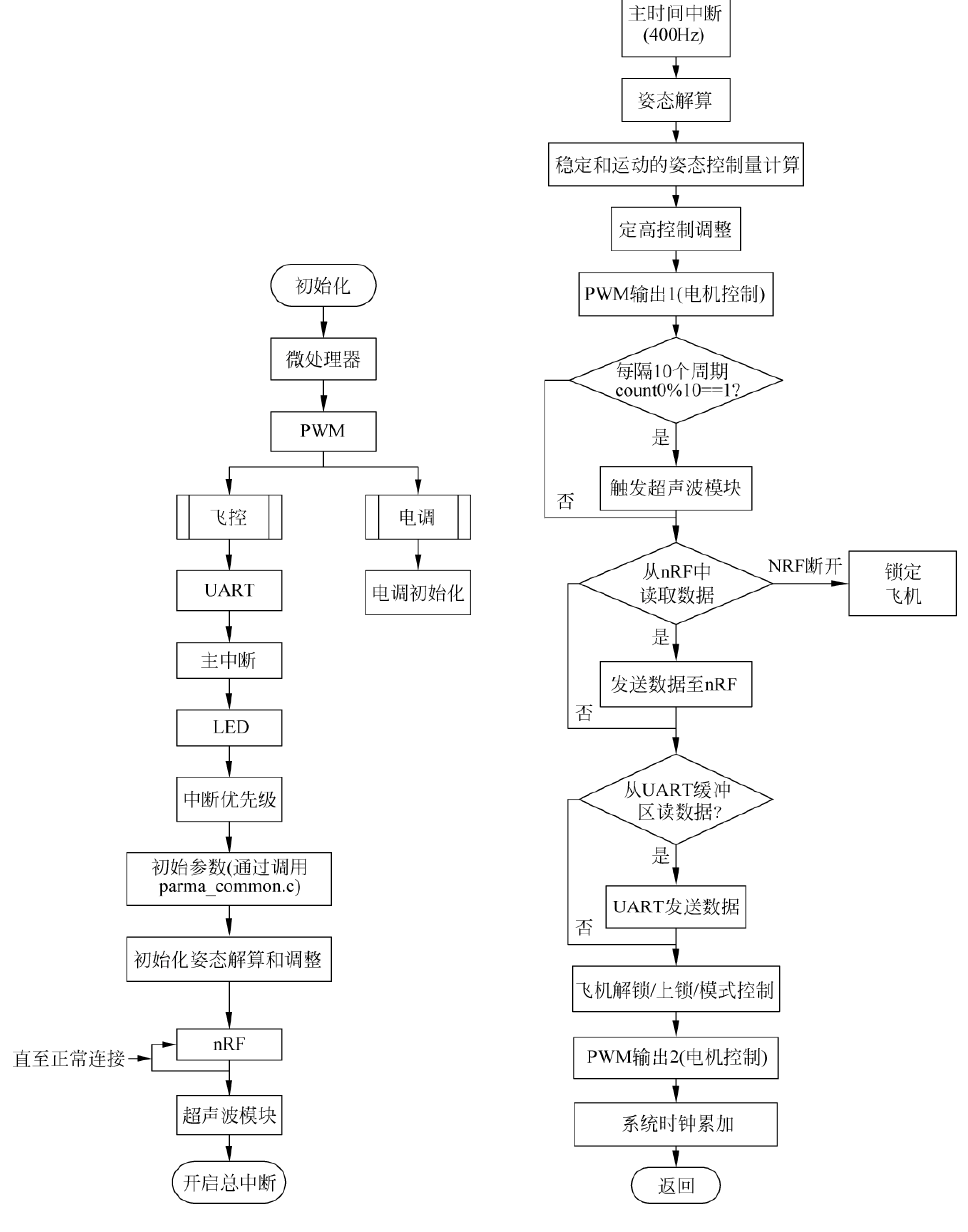


图 5.2.1　初始化流程框图

图 5.2.2　主中断流程框图

表 5.2.1　重要变量列表

变　量　名	变 量 含 义
attitudeActual. Roll	实际横滚角
attitudeActual. Pitch	实际俯仰角
attitudeActual. Yaw	实际偏航角
stabDesired. Throttle	油门控制量
stabDesired. Roll	横滚角/速度目标量
stabDesired. Pitch	俯仰角/速度目标量
stabDesired. Yaw	偏航角/速度目标量
stabDesired. StabilizationMode[i]	飞行控制模式
flightStatus. Armed	解锁/锁定状态
actuatorDesired. Throttle	油门输出量
actuatorDesired. Roll	横滚方向输出量
actuatorDesired. Pitch	俯仰方向输出量
actuatorDesired. Yaw	偏航方向输出量
sonar_distance[0]	最近高度值
nRF_Buf_Out. Data_int[i] /nRF_Buf_Out. Data_char[i]	nRF 输出值
nRF_Buf_In. Data_int[i] /nRF_Buf_In. Data_char[i]	nRF 输入值
Uart_Buf_Out. Data_int[i] /Uart_Buf_Out. Data_char[i]	UART 输出值
Uart_Buf_In. Data_int[i] /Uart_Buf_In. Data_char[i]	UART 输入值

5.3　四旋翼飞行器具体功能软件实现

5.3.1　定高飞行

定高系统采用超声波获取数据，由遥控输出的油门量控制飞机距离地面的高度。飞机实际的油门量为遥控输出的油门量减去由高度进行 PD 计算所得出的控制量。这样的条件下，当实际油门量所产生的升力刚好与飞机重力相抵消时，飞机达到平衡状态。而每当飞机高度变化或遥控输出油门量变化，飞机都能通过上升或下降达到新的平衡。

然而这种方法各挡油门所确定的高度与电池电量、飞机配重等密切相关，不能精确定高，因此后续设计中，更宜采取将遥控输出量直接对应相应的高度，根据飞机高度与设定高度的差直接进行定高控制，定高系统将更加精确和可靠。

由于飞机的姿态始终在变化，超声波获取的高度数据并非真实的高度数据，因此需要根据飞机的横滚角和俯仰角对超声波数据进行修正。若超声波模块处于飞机的正中心，那么实际高度 $h'=h\cos\alpha\cos\beta$(α、β 分别为横滚角和俯仰角)，如果超声波模块不在飞机

的正中央，那么实际高度设为 h'，则 $h_1 = h\cos\alpha \pm a\sin\alpha$、$h' = h_1\cos\beta \pm b\sin\beta$，$a$、$b$ 取决于超声波模块与飞机中心的距离，正负取决于超声波模块与飞机中心的相对位置。

在获取高度后，并不直接覆盖上次获取的超声波数据，而是将上一次获取的超声波数据存入旧高度值变量，再将新的数据存入当前高度值变量，两次的数据可以用来计算飞机高度方向的速度值，用于 PD 计算。定高系统设计框图如图 5.3.1 所示。

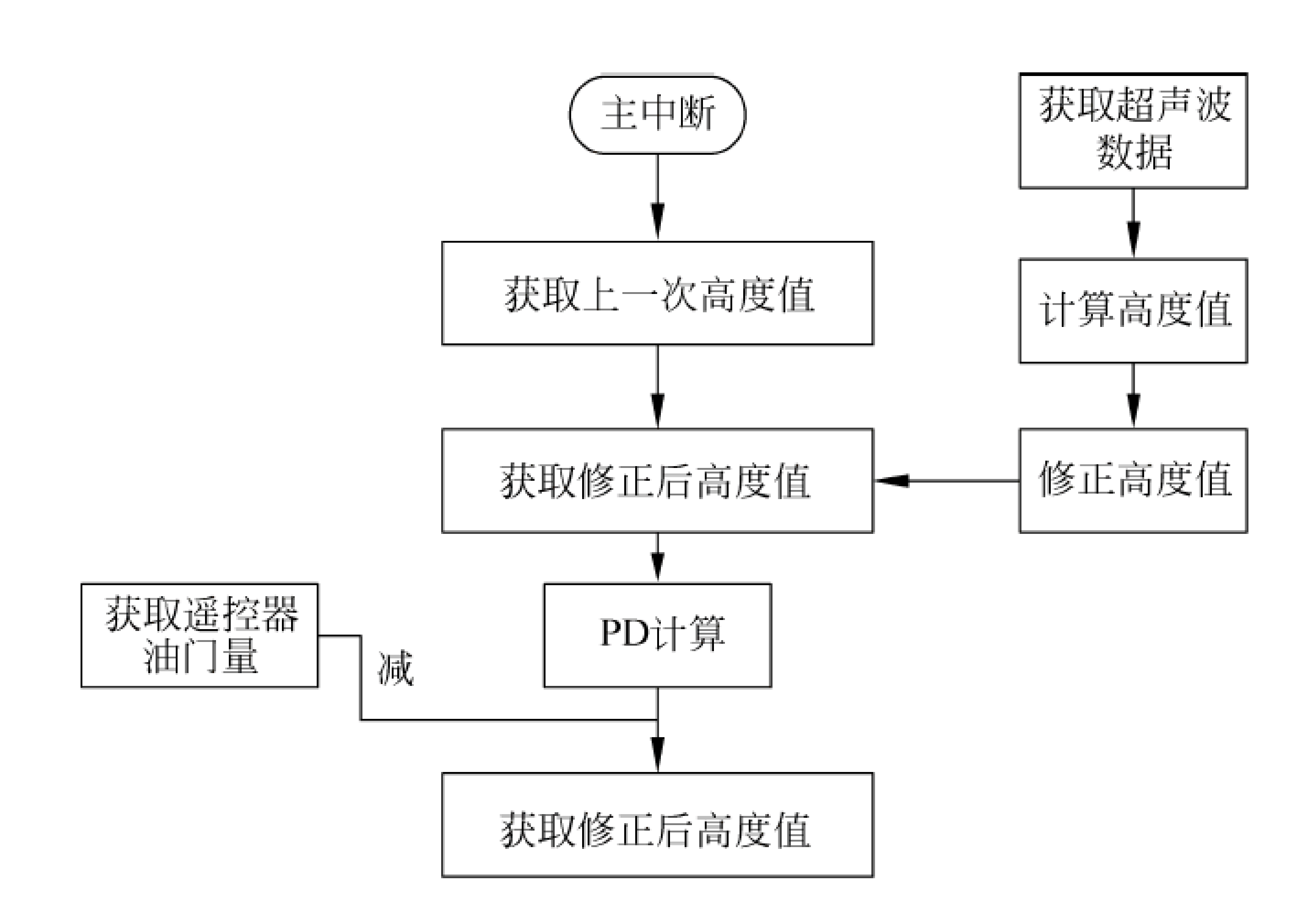


图 5.3.1 定高系统设计框图

5.3.2 定点悬停与位移控制

定点悬浮是在姿态控制及高度控制的基础上实现的，首先姿态控制使飞机能够平稳地飞行，而高度控制是飞行器保持在目标高度上，最后通过光流传感器获取飞行器的位移信息，不断地调节飞行器的倾斜角度，使飞行器保持在固定一个点附近，实现定点悬浮的功能。定点悬浮流程框图如图 5.3.2 所示。

ADNS3080 光流软件编程如下（设定高度为 300mm）：

```
if ((High_Now > 300)
&&((ADNS3080_Data_Buffer[0] & 0x80) != 0)
&&(ADNS3080_Data_Buffer[3]>= 10)
&&(att_in->rol <= 45)
&&(att_in->rol >=  -45)
&&(att_in->pit <=  45)
&&(att_in->pit >=  -45))
{
    float diff_roll = att_in->rol - last_roll;
```

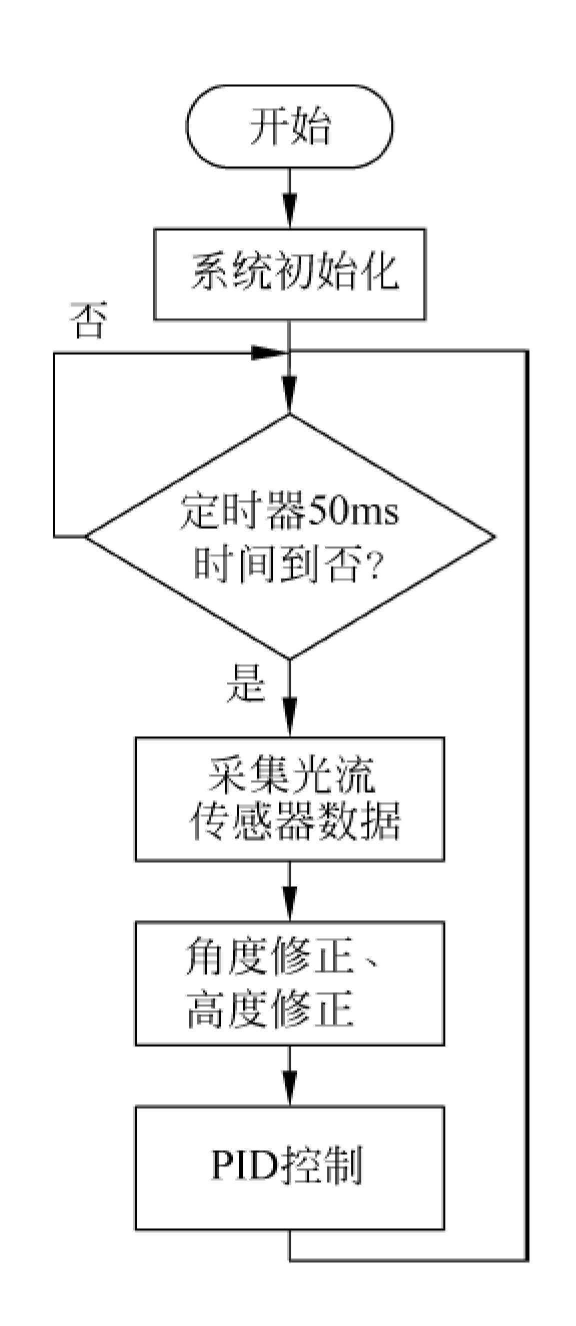


图 5.3.2　定点悬浮流程框图

```
    float diff_pitch = att_in->pit - last_pitch;
    change_x = x_pos - (diff_roll * radians_to_pixels_x);
    change_y = y_pos +(diff_pitch * radians_to_pixels_y) ;
    x_cm = change_x * avg_altitude * conv_factor;
    y_cm = change_y * avg_altitude * conv_factor;
    tot_x_cm += x_cm;
    tot_y_cm += y_cm;
}
pos_x_i += tot_x_cm;
pos_y_i += tot_y_cm;
if(pos_x_i > pos_x_i_max)  pos_x_i = pos_x_i_max;
if(pos_x_i < -pos_x_i_max)  pos_x_i = -pos_x_i_max;
if(pos_y_i > pos_x_i_max)  pos_y_i = pos_x_i_max;
if(pos_y_i < -pos_x_i_max)  pos_y_i = -pos_x_i_max;
PID_POS_X.pout = PID_POS_X.P * tot_x_cm;
PID_POS_X.iout = PID_POS_X.I * pos_x_i;
PID_POS_X.dout = PID_POS_X.D * (tot_x_cm - last_tot_x_cm);
PID_POS_X.OUT = PID_POS_X.pout + PID_POS_X.iout + PID_POS_X.dout;
PID_POS_Y.pout = PID_POS_Y.P * tot_y_cm;
PID_POS_Y.iout = PID_POS_Y.I * pos_y_i;
PID_POS_Y.dout = PID_POS_Y.D * (tot_y_cm - last_tot_y_cm);
PID_POS_Y.OUT = PID_POS_Y.pout + PID_POS_Y.iout + PID_POS_Y.dout;
```

if 条件语句中，先检测飞行器高度是否在 300mm 以上（High_Now > 300，光流传感

器由于焦距原因只能测量距离 30cm 以上物体表面的光流运动)，光流传感器是否有运动(ADNS3080_Data_Buffer[0] & 0x80)，并且飞行器的 roll、pitch 倾斜角度不能超过 45°才会进行光流位移运算。

if 语句中的前四句代码用于修正飞行器倾斜时给光流传感器测量值带来的误差，飞行器转动时光流传感器也会测量到有光流运动，而在定点悬浮中需要的平移位移，需要将这些倾斜造成的位移去除掉，具体如图 5.3.3 所示。

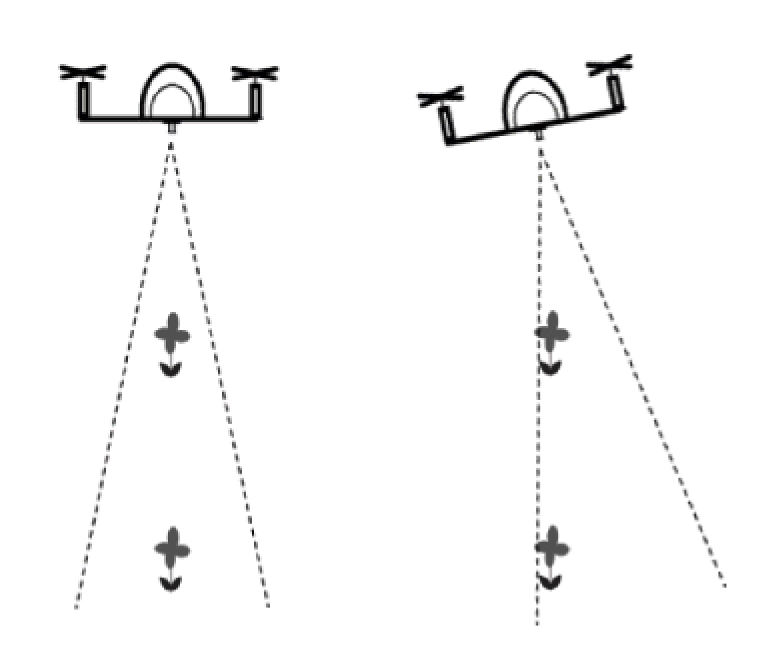


图 5.3.3 光流传感器角度修正示意图

飞行器 roll 角度方向转动造成的光流位移误差值计算公式为

$$X_{\text{exp}} = \frac{\delta_r \cdot \text{Res} \cdot s}{F} \tag{5.3.1}$$

式中，X_{exp}为 X 的估计值；δ_r为横滚角度的改变值；Res 为传感器分辨率(像素)；s 为比例系数；F 为传感器视野。上面的代码中 diff_roll 和 diff_pitch 是飞行器的转动误差角度，radians_to_pixels_x 和 radians_to_pixels_y 分别为两个方向角度修正的系数，diff_roll * radians_to_pixels_x 和 diff_pitch * radians_to_pixels_y 分别代表飞行器在 roll 和 pitch 方向转动时造成的光流位移误差值，最后要从 x_pos 和 y_pos 中去除这个误差。

高度不同，飞行器移动相同的距离时光流传感器测得值有所不同，因此还要进行高度修正。原理图如图 5.3.4 所示。

高度修正公式为

$$d = \frac{vH}{\text{Res} \cdot s} \times 2.0 \times \tan \frac{F}{2.0} \tag{5.3.2}$$

式中，d 为移动的距离；v 为传感器输出值；H 为高度，代码中 change_x 和 change_y 对应光流传感器的位移值，avg_altitude 是当前飞行器高度单位为 cm，conv_factor 为转换系数，最后得的真实飞行器平移值单位也为 cm。

后面两句代码将每次光流传感器测量到的位移值累加进总位移值 tot_x_cm 和

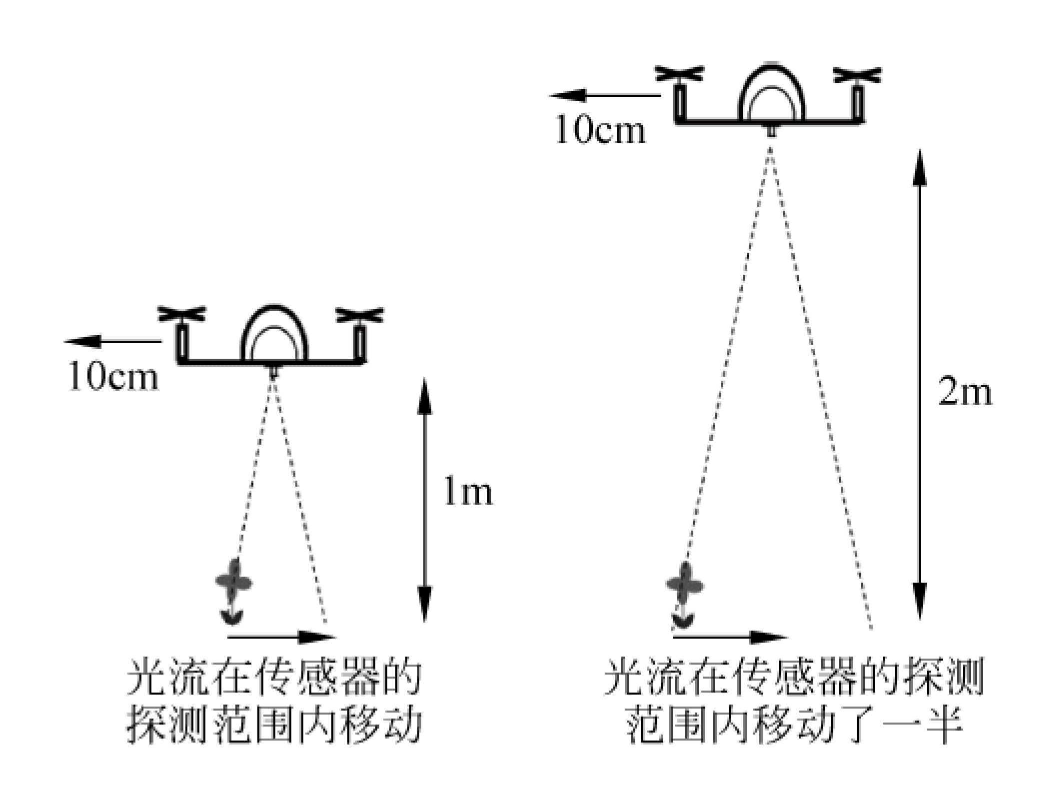


图 5.3.4　光流传感器高度修正示意图

tot_y_cm 中，得到飞行器在 X、Y 方向上与原点的真实距离值。

接下来的代码是飞行器定点悬停的 PID 调节代码，最后定点悬停的输出量 PID_POS_X.OUT 和 PID_POS_Y.OUT 控制的是飞行器的 U_2（翻滚输入控制量）、U_3（俯仰控制量），使 tot_x_cm 和 tot_y_cm 的值趋向于 0，则飞行器能够保持飞行在原点附近，实现定点悬停的功能。

除了 ADNS3080 光流传感器外，还常采用 PX4Flow 光流，以下介绍一些其独特之处。

PX4Flow 采用 MAVLINK 协议传输，传输的基本单位是消息帧，一帧的数据长度为 8B 到 263B 不等。

mavlink 消息帧里最重要的两个东西：一个是 msgid；另一个是 payload。前者是 payload 中内容的编号，后者则存放了消息。消息有许多种类型，在官网的主页中以蓝色的“#”加数字的方式来表示消息的编号，如“#0”（这样的表示方法应该是为了方便在网页中查找相应编号消息的定义）。下面将以几个消息为例，讲解 mavlink 消息。

具体分析一个心跳包：

FE	09	12	01	01	00	(00	00	00	00	02	03	51	03	03)	7B	7A

FE 为包开头，09 为数据包长度，12 为消息序号，01 为硬件系统序号，第五位是消息类型序号，括号里面是有效数据载荷（长度为 9），最后两位是数据校验位。通过消息类型序号可以看到这是 0 号消息，0 号消息是心跳包消息。重点分析一下这个长度为 9 的有效载荷，心跳包里面有 6 个成员变量。

因此需要什么数据，了解其消息类型序号，在执行一个串口接收程序即可。

位移控制也是在姿态控制及高度控制的基础上实现的，位移控制与定点悬浮都用到了光流传感器，位移控制是在有位移命令的情况下才执行的。飞行器首先处于定点悬浮的状态，当用户设定飞行器向任意方向移动任意距离时，飞行器进入位移控制状态，通过PID控制器不断地调节飞行器倾斜方向的角度，使飞行器平稳、快速地向目标地点飞行，同时用光流传感器测量移动的距离，当达到目标地点后，退出位移控制模式，再次进入定点悬浮模式。程序流程框图如图5.3.5所示。

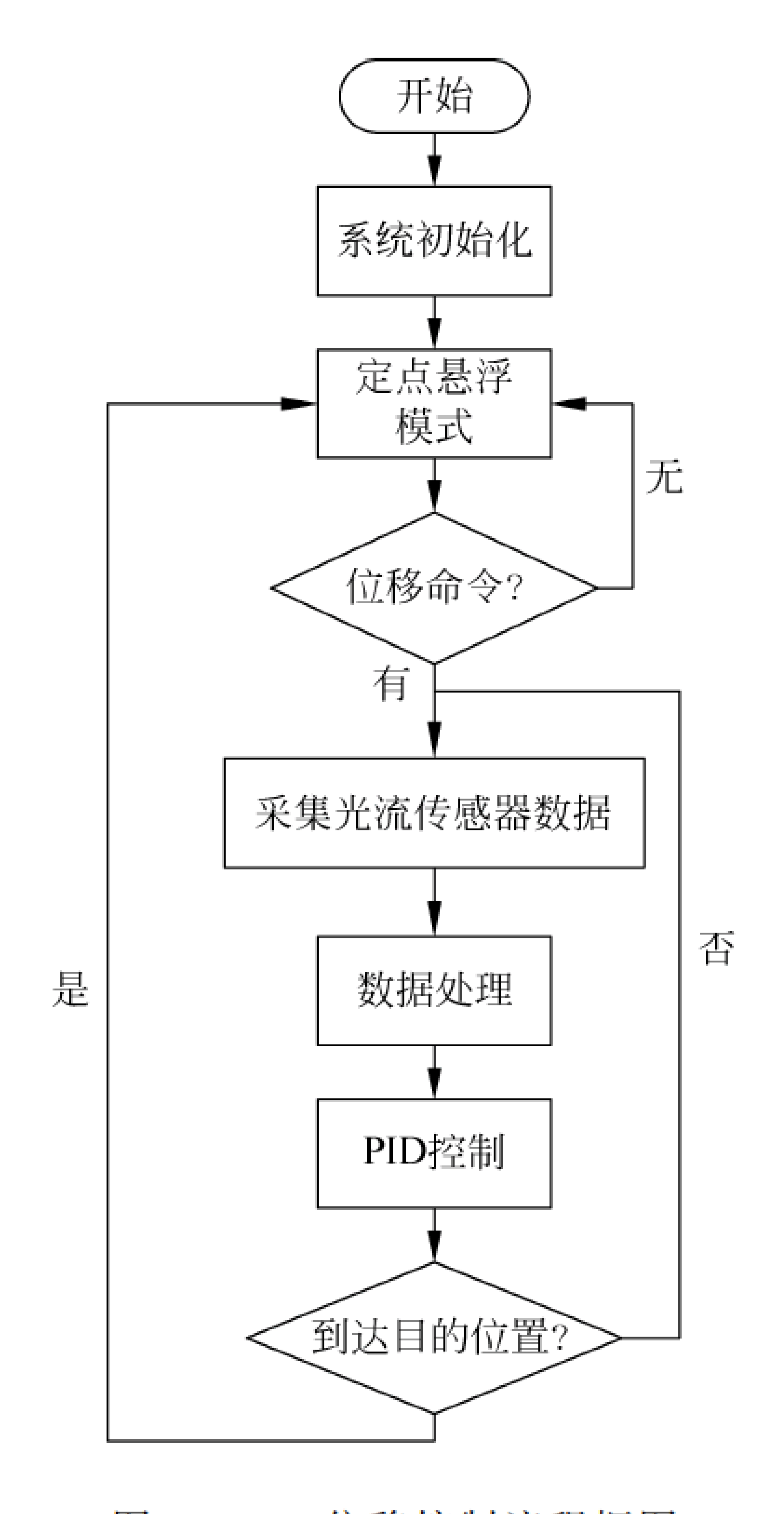


图5.3.5　位移控制流程框图

需要注意的是，如果只设定一个方向的位移时，需要保持另一个方向的位移不偏离，即一个方向做位移PID调节，另一个方向做定点PID调节。位移控制的调节量控制的是飞行器的U2(翻滚输入控制量)、U3(俯仰控制量)，使tot_x_cm和tot_y_cm的值趋向于设定的目标位移值，飞行器飞向目标位置，实现位移控制的功能。具体的PID调节部分代码和定点悬浮的代码类似，在此不再赘述。

5.3.3　数据传输设计

1. 数据传输方式

1）UART 传输

UART 传输为保证一帧数据完整对于同步性的要求比较高，所以在传输时需要加入同步位和同步程序。接收数据时，需要运行同步程序，验证完事先设定好的数位同步位之后，再从缓存区中获取接收到的有效数据，发送数据时则要在发送的数据前加上同步位。

缓存区的使用，降低 UART 传输对于同步性的要求，使 UART 传输更加灵活，也使飞行控制程序的设计更加便捷。

在 TI 系列 ARM 芯片中，有 10 余个 UART 传输模块，根据微处理器手册，可以找出不同 UART 模块对应的端口，即可配置相应端口功能、中断和波特率等。

2）nRF 传输

nRF 作为一块集成电路，是作为微处理器的下位机使用，只需要通过微处理器对其做相应的初始设定，就可以接收由其他无线模块发送的数据并存储在 nRF 上的缓存区中。微处理器可以在适宜的时间从 nRF 的缓存区中获取数据。当微处理器想要向外发射数据时，将 nRF 设定成发送模式，就可以将微处理器中的数据通过 nRF 发送出去。

nRF 与飞行控制板之间采用的是 SPI 传输模式，这种模式速度较快，适应了数据即时交换的要求。虽然 SPI 是一种非常实用的双工传输模式，但是由于 nRF 不支持双工通信，因此程序中采用了伪双工模式通信。

nRF 的数据传输每次传输 8 位，而 8 位的数据不能传输更加精确的飞行控制数据，因此需要将 16 位的飞行控制数据转换为 8 位进行传输。程序中在一个 Union 中定义了 char 类型变量和 16 位 int 类型变量，由于两个变量公用内存空间，因此编辑发送的数据时采用给 int 类型变量赋值，而发送时将 char 变量发送出去，简单地实现了数据位的拆分，也方便了发送不同的飞行控制数据，便于调试。

2. 数据帧设计

为了数据能够准确传输，在两个设备进行通信时需要设计合理的数据帧格式，包含帧头、帧类型位、所传数据及校验和。

帧头用来识别一个帧的开头；帧类型位用来表示所传数据类型，实现帧的多功能复用；所传数据必须与帧类型位所表示数据类型一致；校验和为校验位之前所有比特之

和，用于判断所传数据是否出错，接收到一个数据帧之后，首先计算除校验位外所有字节之和，并与校验位比较，如果相同，则该数据帧可以使用，若不同，则丢弃该帧。

数据帧能否正确传输直接关系到无人机飞行的稳定性和安全性，因此在数据帧传输时需遵循"宁缺毋滥"的原则，保证数据正确传输。

以下为数据帧设计实例，为上位机和地面站数据帧格式。

```
上位机与地面站通信的一帧数据包含 32B:
帧头 ---------------- 2B
        0x9C 0x5F
帧数据类型 ---------- 1B
(上位机发往地面站时采用)
    0x23 ------- 需要转发的 PID 参数(Pitch/Roll/Yaw)
    0x24 ------- 需要转发的 PID 参数(Height/Pos_X/Pos_Y)
    0x25 ------- 询问 PID 参数
    0x91 ------- X 方向偏置
    0x92 ------- Y 方向偏置
    0x93 ------- 偏航角偏置
(地面站发往上位机时采用)
    0x1c ------- 遥控器数据
    0x02 ------- 飞行器姿态数据
    0x04 ------- 飞行器上存储的 PID 数据(Pitch/Roll/Yaw)
    0x05 ------- 飞行器上存储的 PID 数据(Height/Pos_X/Pos_Y)
数据 ---------------- 28B
校验和 -------------- 1B
```

5.3.4 控制系统设计

控制系统主要由 3 个部分组成，接收遥控器或设定的数据、通过 PID 运算计算出合适的输出量、将输出量通过 PWM 波输出，每个部分缺一不可。

接收的遥控器的数据包括飞机的模式数据和 3 个姿态角的控制量，同时要进行解锁/锁定的处理。若飞机处于锁定状态(DISARMED)，而遥控器发出了解锁指令，则飞机要经过 200 个主时钟周期完成解锁。若飞机处于解锁状态(ARMED)而遥控器发出锁定指令，则在 190 个主时钟周期之后完成锁定。这 200/190 个周期给飞机赋予的状态为正在解锁/锁定(ARMING)，在这个状态下，飞机根据遥控的指令继续解锁或锁定。若遥控指令与飞机状态相同，则不进行解锁和锁定操作。

飞行控制板的输出量则是经由 openpilot 库中的 stabalization.h 函数，根据程序设定

好的模式(rate、attitude 或其他)进行一层或两层 PID 运算,获取相应的初始输出量。

经过稳定性计算,计算得到油门量 t、俯仰控制量 p、横滚控制量 r 和自旋控制量 y,根据以下公式计算出 4 个电机的输出量:

```
m1 = t + r - p + y;
m2 = t - r - p - y;
m3 = t - r + p + y;
m4 = t + r + p - y;
```

计算所得的 4 个电机的输出量,则要通过微处理器的 PWM 波输出功能输出到电调,但在此之前,还需要对输出量进行检验,若超出实现设定好的输出量的上下限,则按照上下限输出,这也是出于对飞机和人的安全以及飞机耐用性的考虑。

PWM 输出使用板载的 PWM 端口输出 PWM 波,Tm4C123G 芯片上有大量 PWM 输出通道,使用其中 4 个 PWM 输出通道,设定周期为 2.5ms,脉宽为 0～2ms,根据无人机飞行状态和控制指令实时改变脉宽以控制电机转速,实现无人机的平稳飞行和设定飞行动作。

在控制系统的设计中,还需要根据 MPU6050 的安放位置设立正确的坐标系,严格按照捷联惯性制导的规则,才能让飞机获取正确的角度值,做出正确的俯仰、横滚和自旋动作维持飞机的稳定或者实现各种自定义动作,如图 5.3.6 所示。

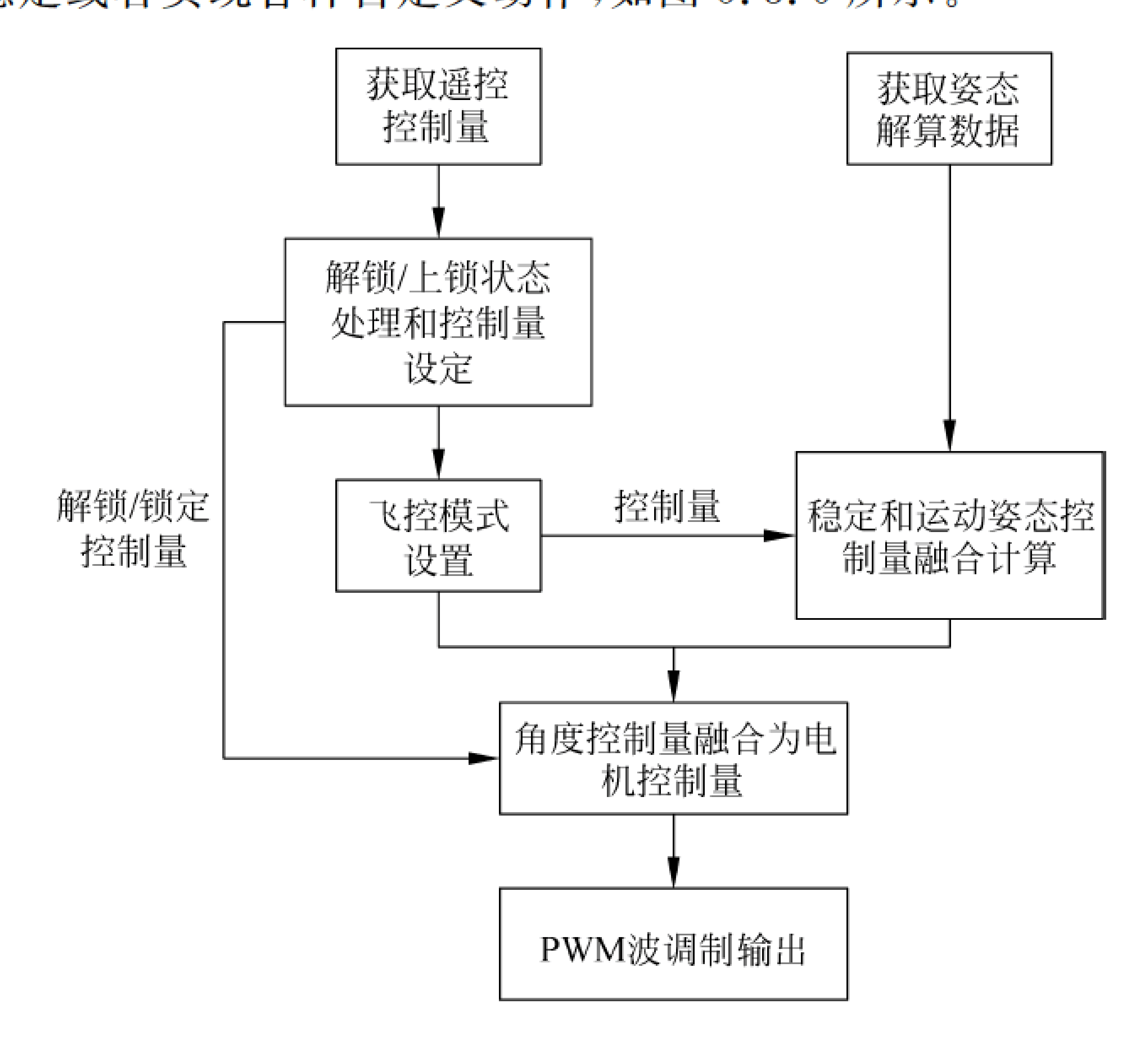


图 5.3.6　控制系统流程框图

5.4 地面站软件设计

在飞行器研发试验过程中，需要实时监控各种数据值的变化。地面控制板软件相当于一个通信的中转站，同时与飞行控制板、遥控器和PC上位机软件通信，将遥控器命令及上位机的命令数据发送给飞行控制板，同时又将飞行控制板反馈回来的实时飞行数据显示在OLED屏幕上或者转发给上位机软件动态显示，要保证这些通信任务相互之间不会发生冲突，也需要同飞行控制板一样有严格的时序控制，具体的软件框图如图5.4.1所示。

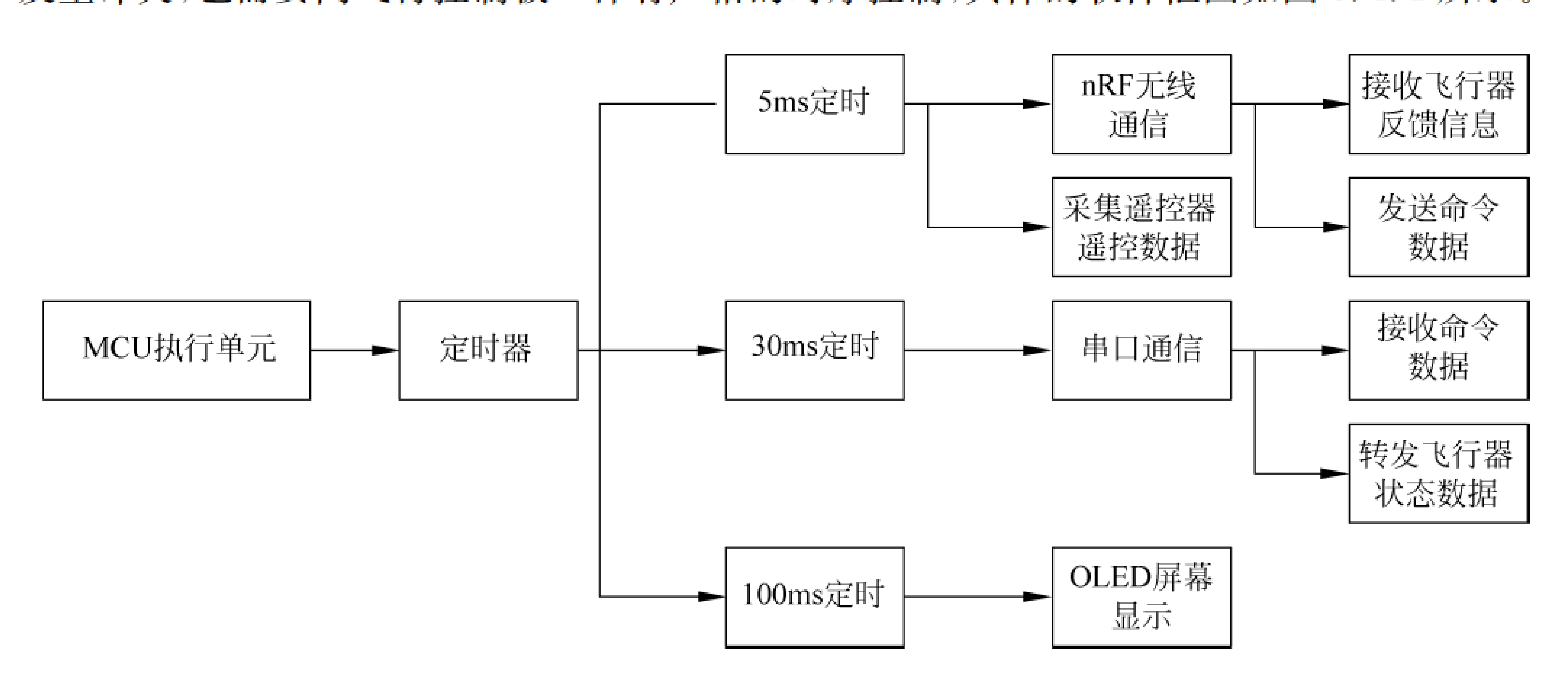


图5.4.1 地面控制板软件框图

5.4.1 地面站系统组成与图形化的界面操作设计

地面站系统LCD(如图5.4.2所示)上显示的内容以下。

① 显示遥控数据，通过引脚中断和时钟计数，测出遥控器传过来的数据。数据格式为：第一个数据是帧头，检测到帧头后，再把后面相应的数据保存在相应的数组中。没有检测到则不保存。显示的数据分别为高度油门值、横滚油门值、俯仰油门值和偏航油门值。

② 显示四旋翼飞机传回来的数据。通过SPI读取nRF的数据包，数据分别为俯仰角(pitch)、横滚角(roll)、偏航角(yaw)、飞行高度(height)、飞机当前坐标(x,y)即光流值。

③ 显示地面站和四旋翼飞机的解锁状态，分别为飞机是否解锁、nRF与地面站的通信是否正常和遥控器与地面站连接是否正常。

④ 波形显示模块。将最近的飞行状态绘制成曲线显示。选择分别有高度、横滚、俯仰和偏航4种，如图5.4.3所示。

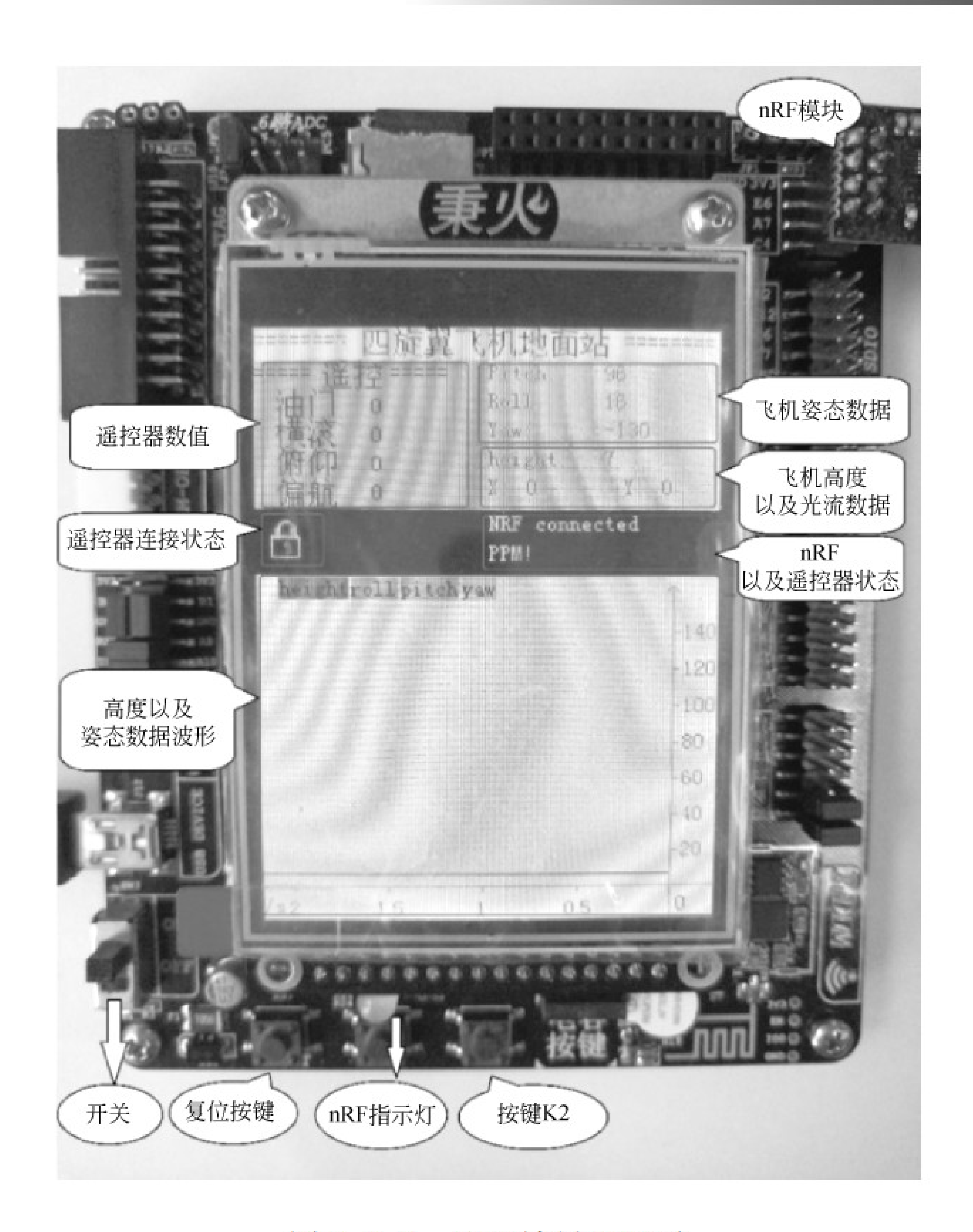
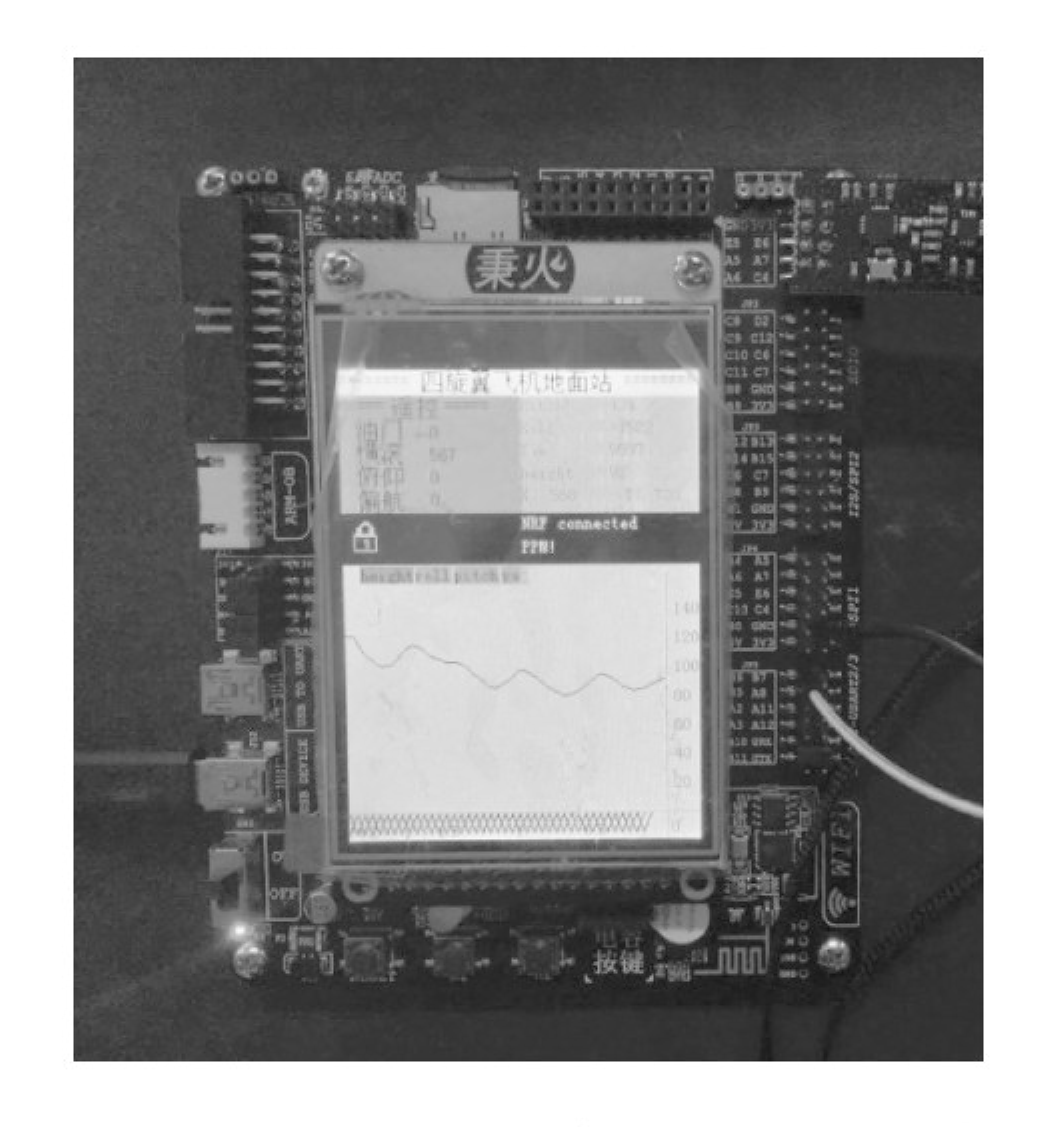


图 5.4.2 地面站界面显示



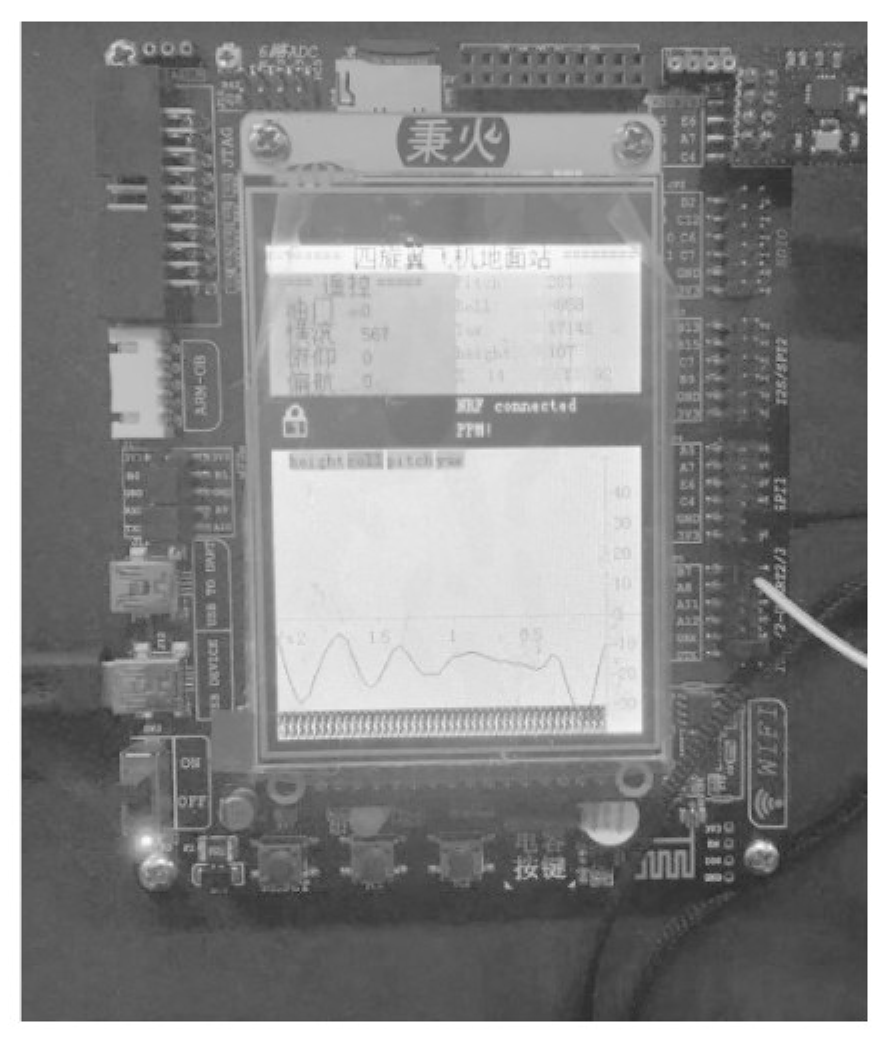


图 5.4.3 高度波形(左)和姿态角波形(右,以 Roll 角为例)显示

5.4.2 与四旋翼飞行器的通信协议

为了实现飞行控制板、地面控制板以及 PC 上位机三者之间的稳定、可靠、快速地进行通信，地面站与上位机使用了特定的数据帧格式进行通信，地面站与飞机是通过 nRF 进行无线通信的。地面站与遥控器也是通过特定的数据帧格式进行通信。

地面站通过识别帧头来判断数据的类型，从而做出相应的反应，并且将末尾等于所有位的累加和作为校验位，来确保数据传输的准确性。数据帧结构如表 5.4.1 所示，地面站与上位机通信协议如表 5.4.2 和表 5.4.3 所示，帧数据共 32B。

表 5.4.1 数据帧格式表

数据帧	帧头	数据类型	数据内容	校验位	说明
长度/B	2	1	28	1	一帧数据长度共 32B

表 5.4.2 地面站与上位机通信协议表 I

地面站接收上位机数据	
数据类型	意　　义
0x23	roll 方向平衡 PID 参数值 P 值
	roll 方向平衡 PID 参数值 I 值
	roll 方向平衡 PID 参数值 D 值
	pitch 方向平衡 PID 参数值 P 值
	pitch 方向平衡 PID 参数值 I 值
	pitch 方向平衡 PID 参数值 D 值
	yaw 方向平衡 PID 参数值 P 值
	yaw 方向平衡 PID 参数值 I 值
	yaw 方向平衡 PID 参数值 D 值
0x24	高度 PID 参数值 P 值
	高度 PID 参数值 I 值
	高度 PID 参数值 D 值
	X 方向悬停 PID 参数值 P 值
	X 方向悬停 PID 参数值 I 值
	X 方向悬停 PID 参数值 D 值
	Y 方向悬停 PID 参数值 P 值
	Y 方向悬停 PID 参数值 I 值
	Y 方向悬停 PID 参数值 D 值
0x25	nRF 发送数据包
0x90	测试飞行(高度、前后、左右)

续表

地面站接收上位机数据	
数据类型	意　　义
0x91	向前后飞行控制量
0x92	向左右飞行控制量
0x93	偏航飞行控制量

表 5.4.3　地面站与上位机通信协议表Ⅱ

地面站发送数据至上位机		
帧头	数据类型	意　　义
0x9c 0x5f	0x02	发送飞机的 pitch 值
		发送飞机的 roll 值
		发送飞机的 yaw 值
		发送飞机的高度值
		发送飞机的光流数据 X 方向值
		发送飞机的光流数据 Y 方向值
	0x1c	发送遥控器横滚值
		发送遥控器俯仰值
		发送遥控器油门值
		发送遥控器偏航值
	0x04	roll 方向平衡 PID 参数值 P 值
		roll 方向平衡 PID 参数值 I 值
		roll 方向平衡 PID 参数值 D 值
		pitch 方向平衡 PID 参数值 P 值
		pitch 方向平衡 PID 参数值 I 值
		pitch 方向平衡 PID 参数值 D 值
		yaw 方向平衡 PID 参数值 P 值
		yaw 方向平衡 PID 参数值 I 值
		yaw 方向平衡 PID 参数值 D 值
	0x05	高度 PID 参数值 P 值
		高度 PID 参数值 I 值
		高度 PID 参数值 D 值
		X 方向悬停 PID 参数值 P 值
		X 方向悬停 PID 参数值 I 值
		X 方向悬停 PID 参数值 D 值
		Y 方向悬停 PID 参数值 P 值
		Y 方向悬停 PID 参数值 I 值
		Y 方向悬停 PID 参数值 D 值

表中 pitch 指的是俯仰，roll 指的是横滚，yaw 指的是偏航。PID 指的是比例积分微分调节。

5.4.3 遥控器 PPM 信号捕获

1. 遥控器信号收发原理

无线遥控就是利用高频无线电波实现对模型的控制，原理如图 5.4.4 所示。如天地飞的 6 通道 2.4GHz 遥控器，具有自动跳频抗干扰能力，从理论上讲可以让上百人在同一场地同时遥控自己的模型而不会相互干扰。而且在遥控距离方面也颇具优势，2.4GHz 遥控系统的功率仅仅在 100mW 以下，而它的遥控距离可以达到 1km 以上。

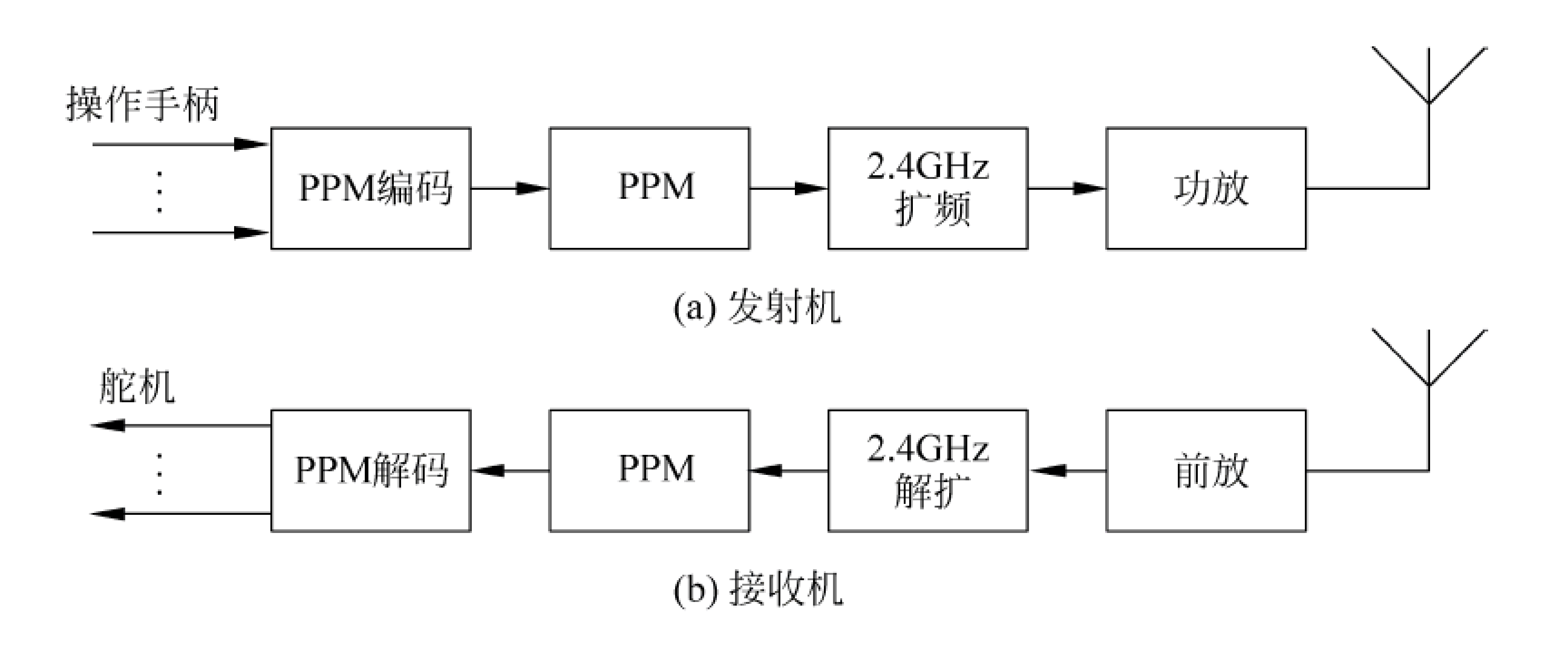


图 5.4.4　遥控器发射机、接收机原理

每个通道信号脉宽为 0～2ms，变化范围为 1～2ms。1 帧 PPM 信号长度为 20ms，理论上最多可以有 10 个通道，但是同步脉冲也需要时间，模型遥控器最多 9 个通道。

2. 读取和解码遥控器信号

由于 PPM 信号通过脉冲相位表示数据，因此地面站读取 PPM 信号中的数据主要通过定时器来实现。

遥控器发出的 PPM 同步信号为高电平，脉冲为负脉冲，因此，当地面站接收到遥控器 PPM 信号时，在上升沿触发开启定时器，在下降沿停止定时器并触发定时器中断，读出定时器 CCR，CCR 的计数值，即反映脉冲与脉冲之间的相位差。同步信号的高电平保持时间较长，因此 CCR 数值较大，可以以此为依据检测同步信号并开始接收数据。同步信号之后获取的新的一组 8 个 CCR 值即为 PPM 信号所传数据，依次存储 8 个 CCR 值，即可解析遥控器信号。

部分实现代码如下：

```
void TIM4_IRQHandler(void)
{
if(TIM4 -> SR & TIM_IT_CC1)       //if(TIM_GetITStatus(TIM2, TIM_IT_CC1) == SET)
    {
        count = TIM4 -> CCR1;     //读 ccr,清 cc1if 标志
        TIM4 -> CNT = 0;          //清 cnt
    }
    ppmdecoder_f();
}

void ppmdecoder_f()
{
  if(5000 < count&&count < 20000)
  {
    ppmChMax = sync;
    sync = 0;
    ppmLoseSig = 0;
  }
  else if(200 < count&&count < 2000&&sync < PPMCH_MAX)
    sync++;
  else
    ppmLoseSig = 1;
  if(ppmLoseSig == 0)
  ppmData[sync] = count;
}
```

5.5　上位机软件设计

上位机软件是使用 C# 语言开发的，负责向用户显示飞行器的实时飞行数据，并提供给用户以图形化的界面操作方式向飞行器发送命令。为了能够快速地通过串口接收数据，并通过各种 UI 控件显示数据，除了主线程外，上位机软件中还建立了两个线程，分别用于接收数据和显示数据。软件框图如图 5.5.1 所示。

5.5.1　软件功能

软件集成的功能包括基本收发功能、飞行控制参数设置功能、光流传感器参数设置功能、RC 遥控器设置功能、飞行控制状态显示功能以及波形显示功能。下面介绍几个主

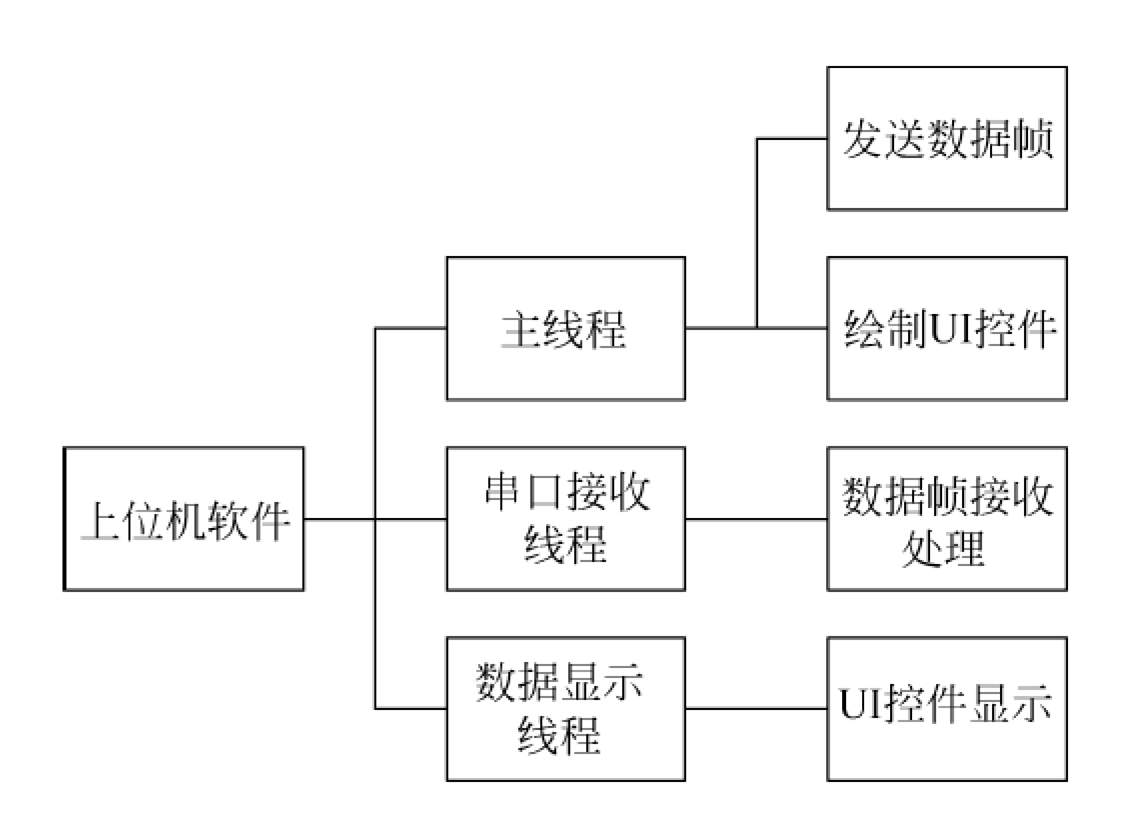


图 5.5.1　上位机软件框图

要功能的特点及使用。

1. 基本收发功能

如图 5.5.2 所示，在基本功能里左侧的大文本框可以输入待发送的字符串或十六进制数据，单击“发送”按钮可以直接发送出去，右侧的大文本框可以显示接收到的字符串或十六进制数据，最右边可以发送自定义的 32B 数据帧，帧头、数据格式以及数据内容都可以自己定义，单击“发送”按钮，数据帧软件会自动加上和校验后发送出去。

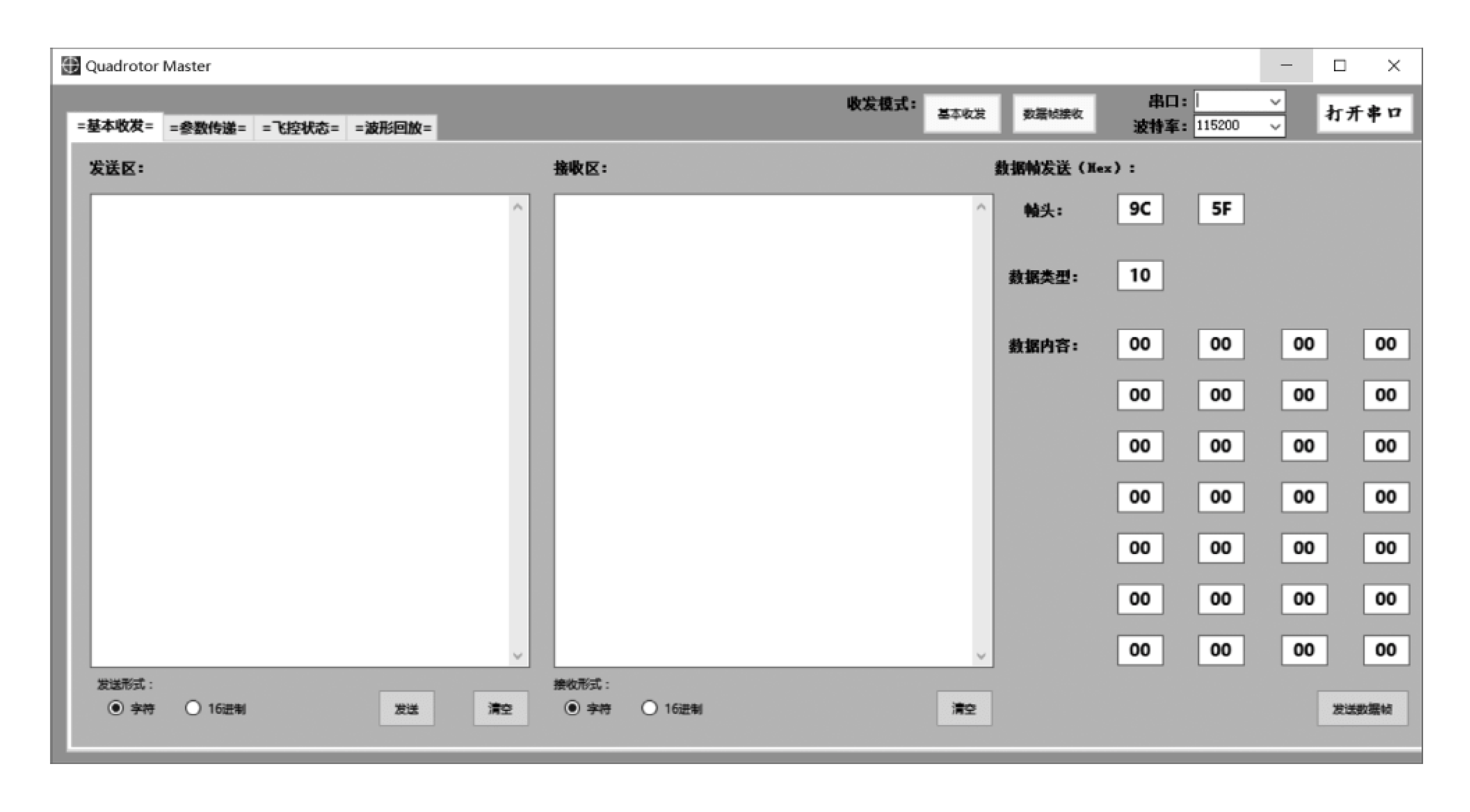


图 5.5.2　基本收发功能截图

2. 飞行控制参数设置功能

如图 5.5.3 所示，在这里可以设置 7 个 PID 控制器的 P、I、D 参数，更改目标高度，修改加速度计和陀螺仪的偏置参数。

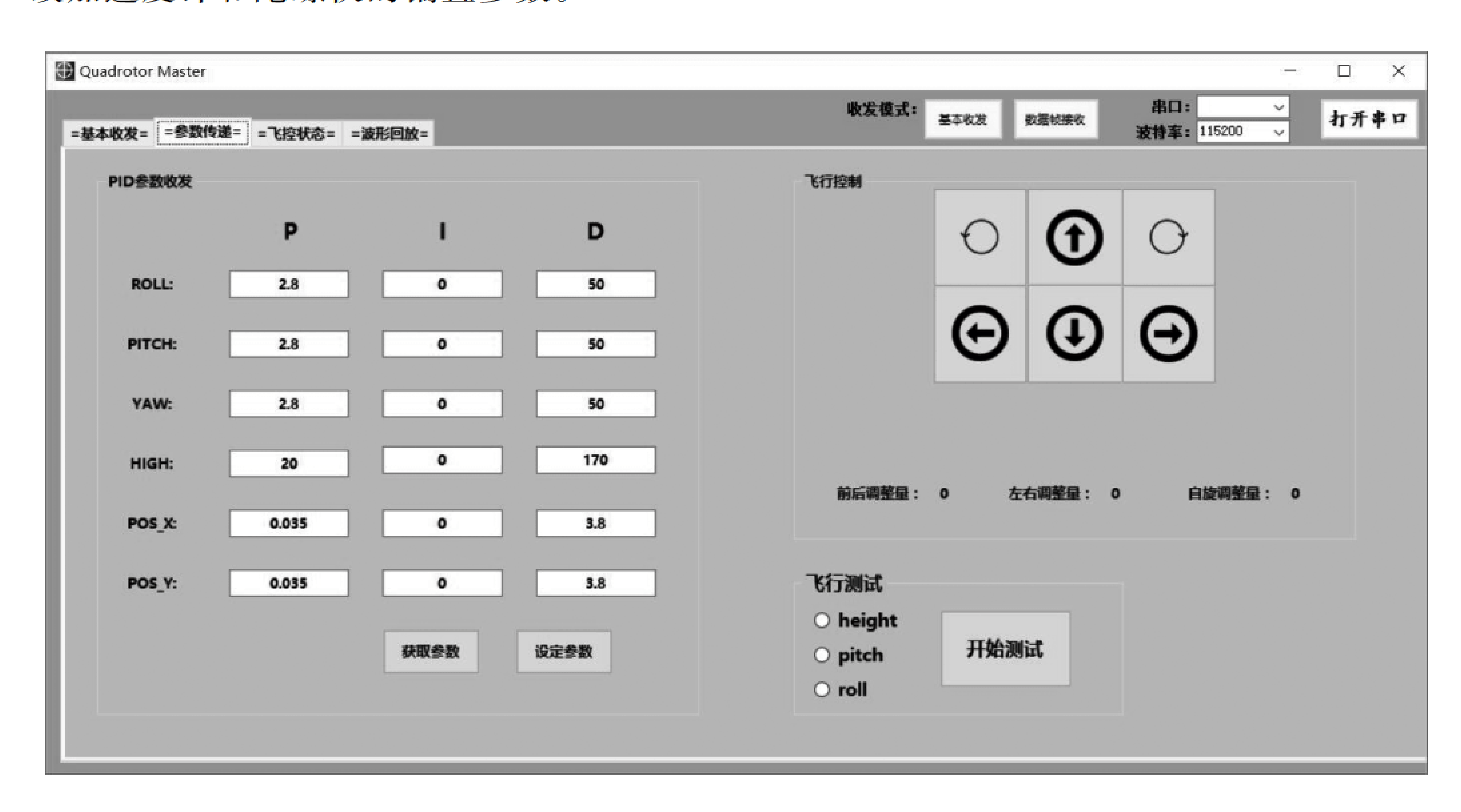


图 5.5.3　飞行控制参数设置功能截图

左边部分为 PID 参数收发，主要用于快速设置飞行器的各种 PID 参数。这部分功能是将设置好的参数写到串口处，单击“设定”参数按钮之后，发送写数据命令，能让读取到数据的地面站将接收到的数据写入飞行控制板的 EEPROM 中，等飞行器断电重启之后，就能获得设定的参数了。

右上部分为飞行控制部分，这部分是基于飞行器悬停位置的操作，具体作用是改变飞行器悬停点，那么飞行器就可以往相关方向飞行。

右下部分为飞行测试模块，这部分是选择将要测试的 radio button，然后单击“开始测试”按钮之后，飞行器会自动进行设定好的轨迹飞行，飞行 10s 之后，飞行器会飞回原来状态，此时会弹出对话框，提示飞行器的测试是否通过以及是否符合标准。如果没能通过，那么会给出建议的 PID 修改方案。

3. 飞行控制状态显示功能

如图 5.5.4 所示，在这个功能里用到了一个使用 OpenGL 开源图形库绘制的动态 3D 模型，可以形象地显示出飞行器的动态姿态，动态模型的右边，Attitude Sensor 组合

框中显示的是飞行器的传感器数据及姿态数据，“遥控器”一栏显示当前遥控器的遥控值，Motor 一栏以进度条的方式显示飞行器 4 个电机的 PWM 值的大小，Other 组合框显示了飞行器高度、位移信息、解锁状态和飞行时间的统计。

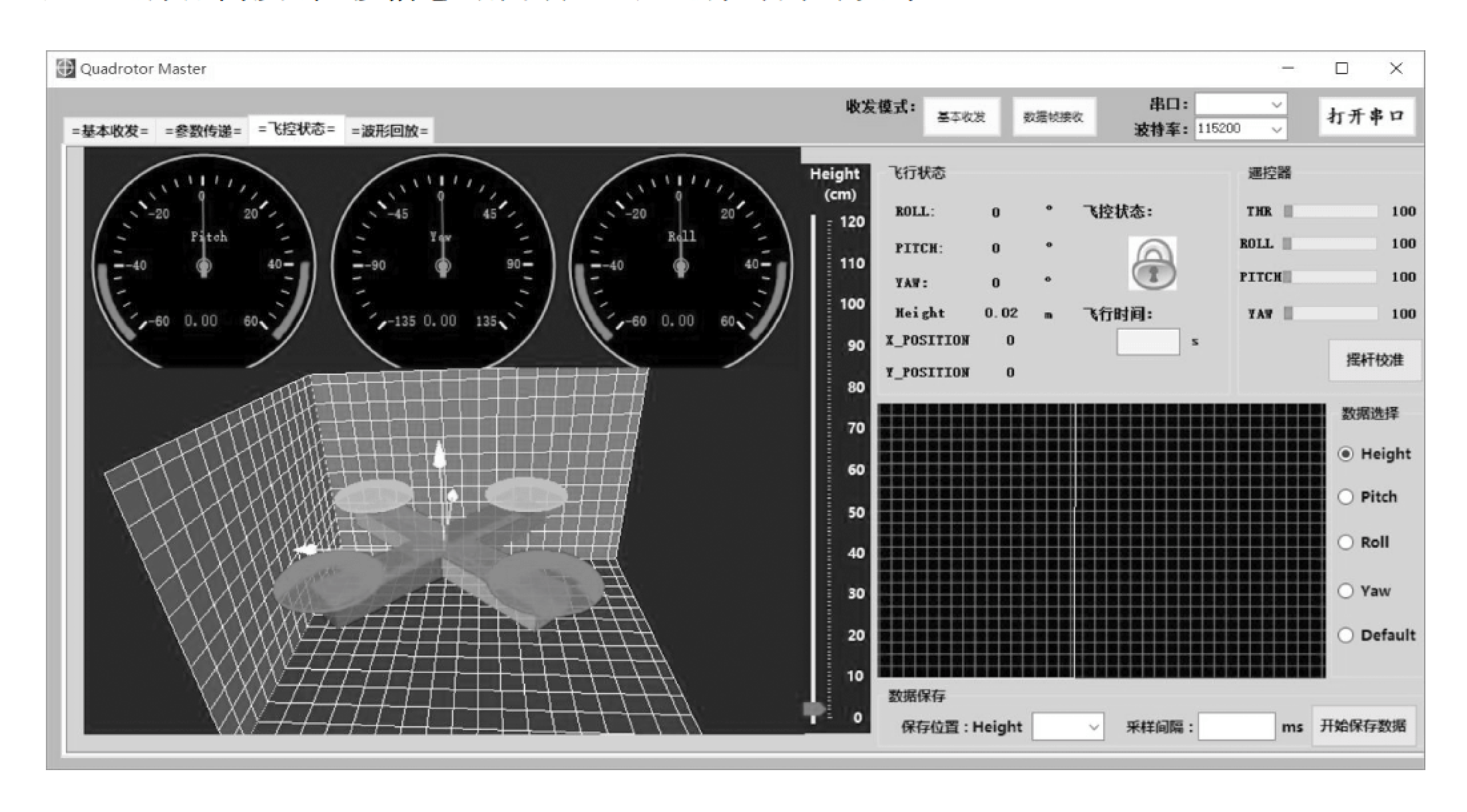


图 5.5.4　飞行控制状态图

进入飞行控制状态选项卡之后，能看到左侧的飞行器模型和仪表盘以及右侧的高度进度条。右边有遥控器的值和下面的 status chart 的显示框。当飞行器飞起来的时候，左侧的框图里面的飞行器会实时地根据真实飞行器的参数进行同步 3D 姿态变化，上面的仪表盘也会指向相应的值。在左右两侧的仪表盘加上了红色区域，目的是提醒飞行器的危险状态。

在右下部分是飞行器数据参数波形显示部分，在这个部分中可以选择相应的数据，那么在飞行器起飞之后，数据就会依次以点连接的方式，在绿色的框图上形成波形，这个时候可以选择保存位置和采样间隔，然后单击“开始保存数据”按钮。那么数据就会实时地保存在 Excel 表格里面，方便以后的查看和使用。

4. 波形显示功能

图 5.5.5 所示功能能够将飞行器的高度数据以波形的方式动态显示出来，以便观察高度变化的规律。

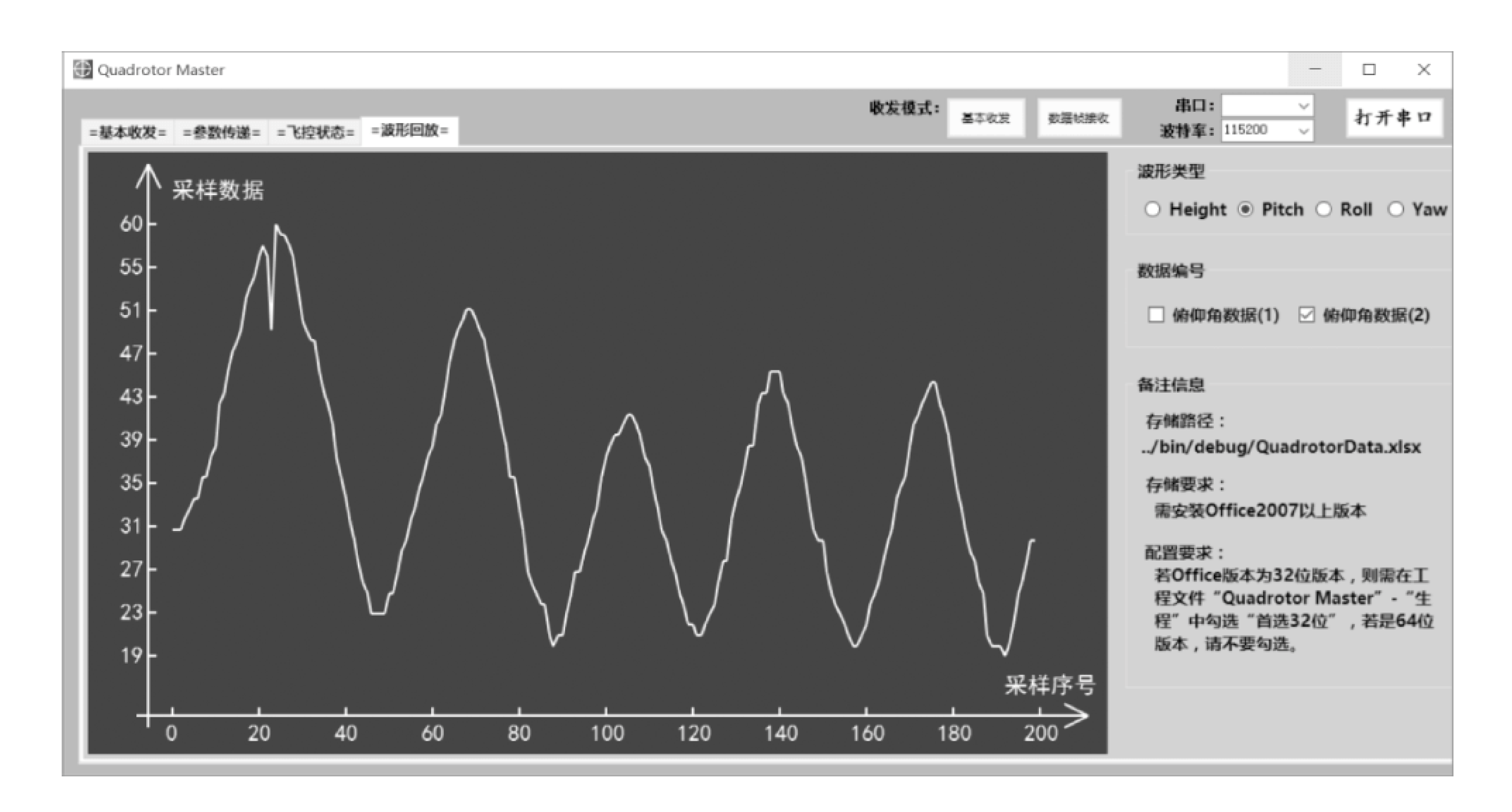


图 5.5.5　波形显示功能截图

5.5.2　软件实现

上位机程序流程框图如图 5.5.6 所示。

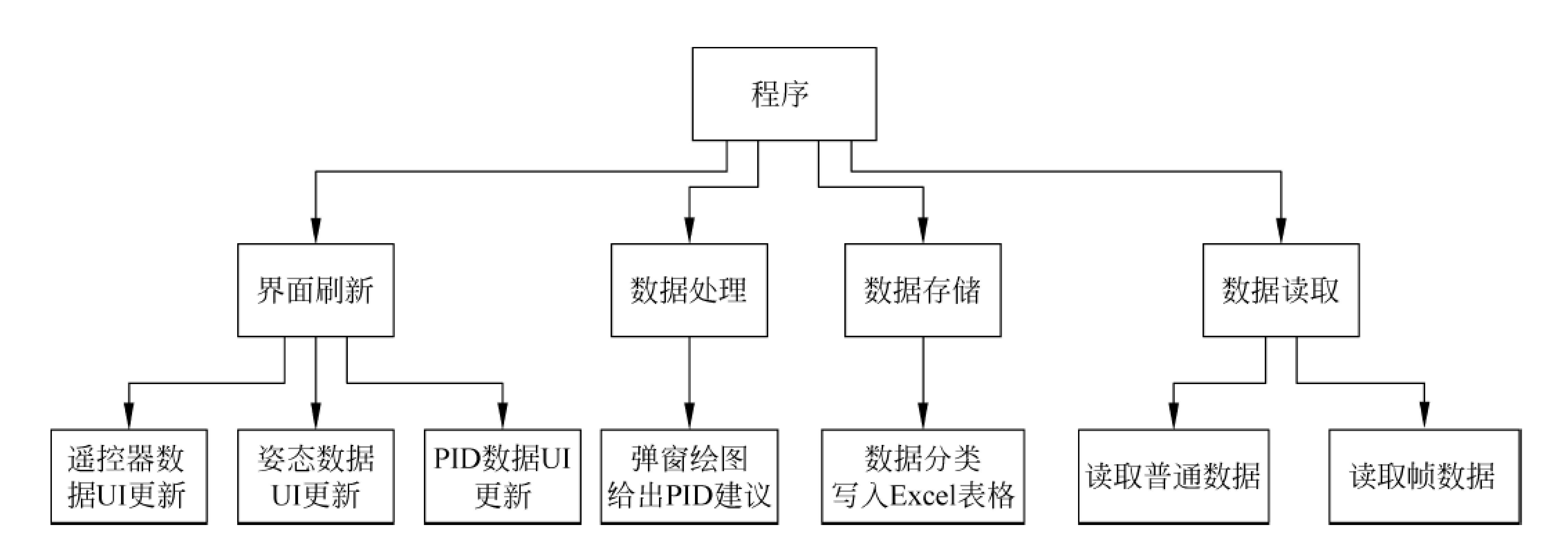


图 5.5.6　上位机程序流程框图

1. 通信模块

上位机软件由 4 个选项卡组成，第一模块为通信测试模块，主要用于串口通信测试，流程框图如图 5.5.7 所示。

通过下拉菜单设置串口端口号和波特率，串口在打开时，首先执行判断操作，如果串

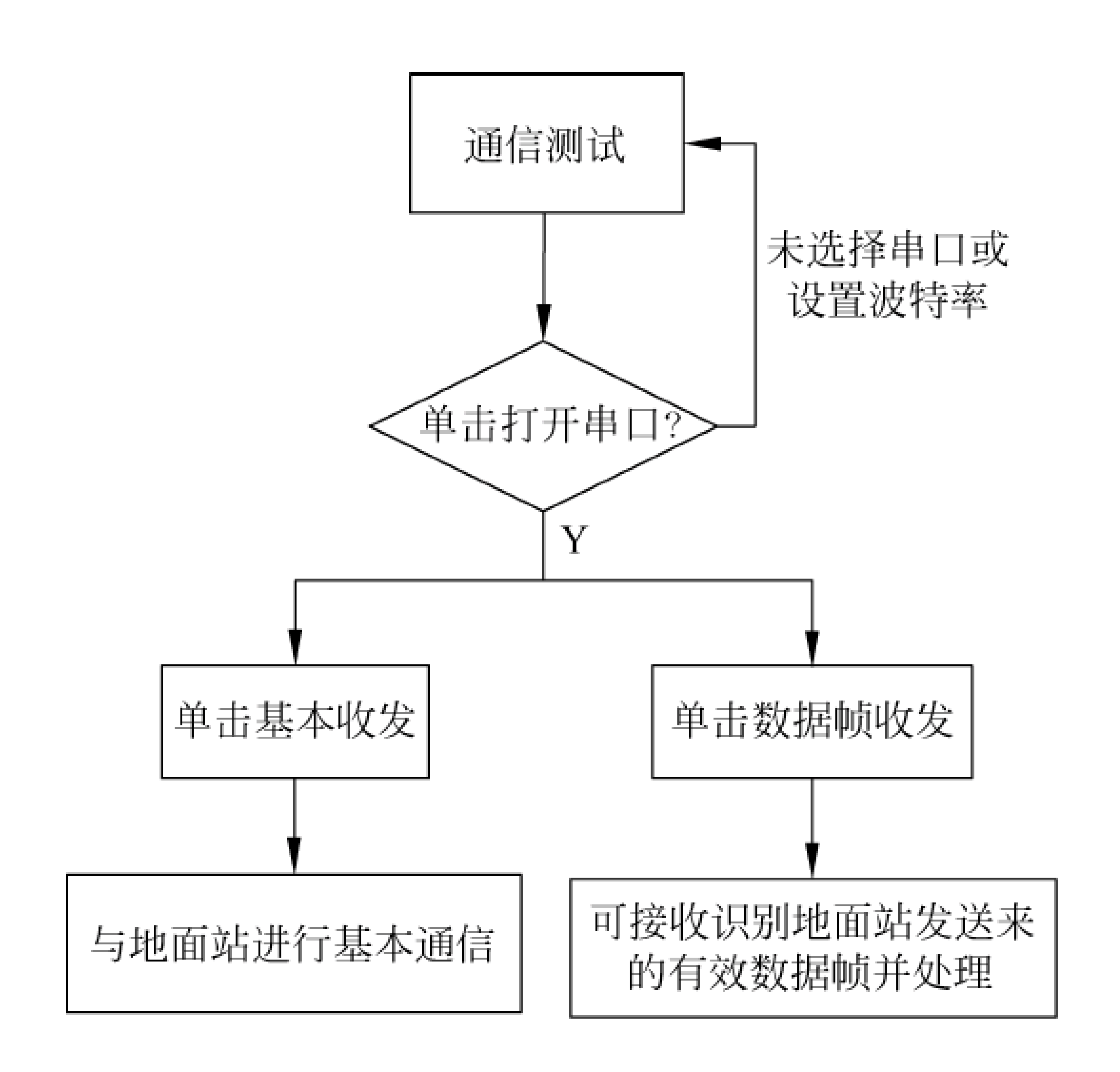


图 5.5.7 通信测试流程框图

口或者波特率没有设置，那么会出现错误提示框，提醒设置串口和波特率。在打开串口之后，可以选择数据收发模式。模式由枚举变量 Mode_Statue 设定：

```
enum Mode_Statue
    {
        None,
        Normal_Mode,
        Frame_Mode,
        Wave_Mode,
        Pic_Mode,
    };
```

常用的为基本收发模式和数据帧模式。

如果选择基本收发模式，就可以使用此界面上的数据发送和清空按钮，通过串口发送数据给地面站，地面站收到后返回发送的数据这一过程来判断数据接收是否正常。

选择数据帧模式之后，就是将带帧头的数据通过串口发送给地面站，地面站通过识别帧头来判断数据的类型，从而做出反应，并且将末尾等于所有位的累加和作为校验位，来确保数据传输的准确性。

一般情况下，选择数据帧收发模式，然后进入参数传递的界面，流程框图如图 5.5.8 所示。

下面介绍这个操作界面上各个控件的作用。

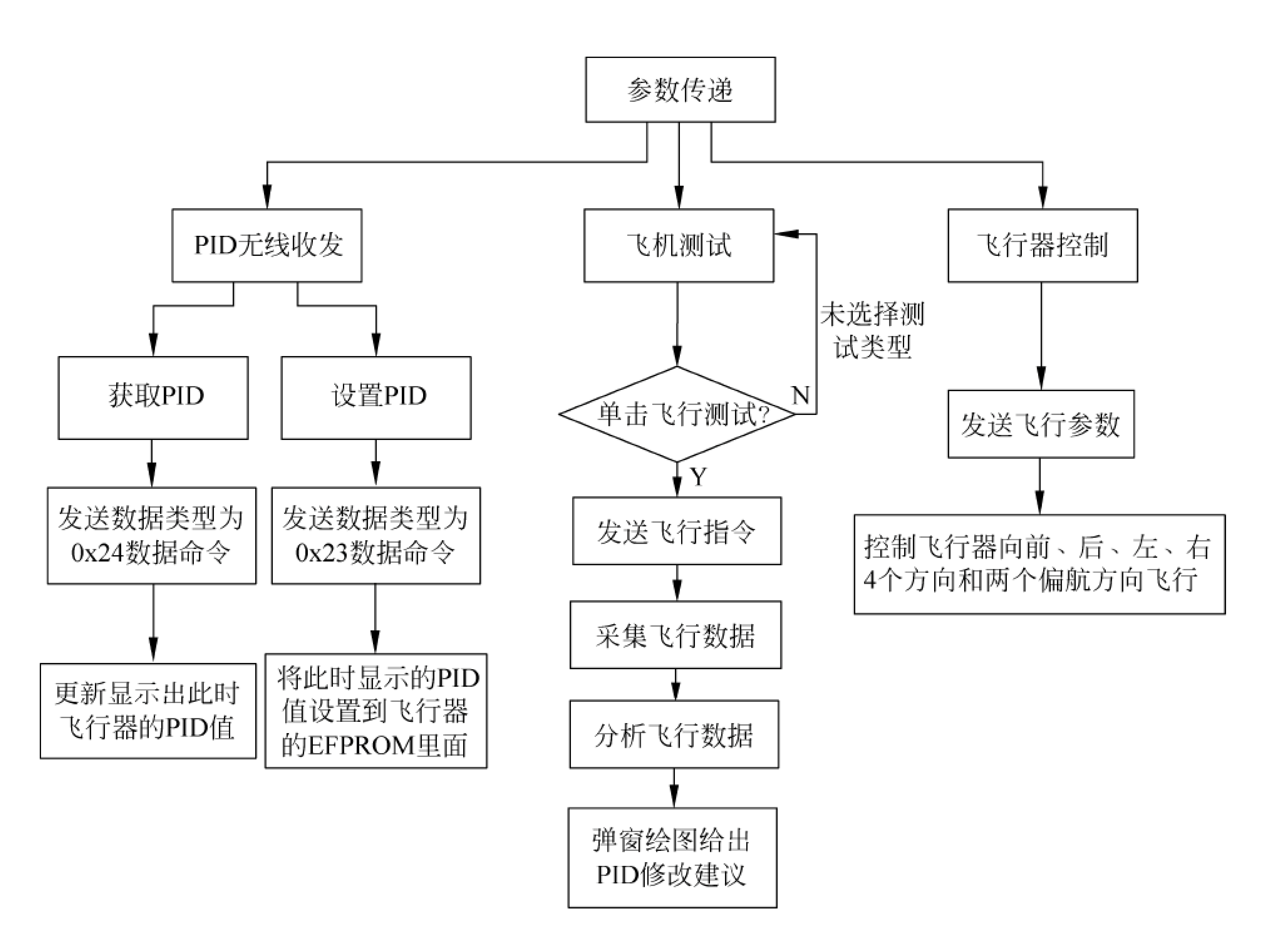


图 5.5.8　参数传递流程框图

1）xtBox 控件

显示此时的 PID 数据。

2）Button 控件

① 获取参数 button：串口写命令给地面站，让地面站读取飞行器此时的 PID 数据并发回到上位机。

② 设定参数 button：将上述所有设定好的参数发送到串口，地面站接收到数据和命令时，将 PID 数据写入飞行器的 EEPROM 里面。

③ 方向控制 button：单击之后设定飞行器的定点位置，实现飞行器的飞行，同时可以调节飞行器的偏航。

④ 开始测试 button：单击之后识别 radio button 的 checked 值，从而进行飞行测试。

Button 发送函数如下：

```
private void button7_Click(object sender, EventArgs e)
    {
```

```
        if (serialPort1.IsOpen == false)
        {
            MessageBox.Show("请先打开串口", "ERROR");
            return;
        }
        updown_ctrl = updown_ctrl + 20;
        label59.Text = updown_ctrl.ToString();
        try
        {
            byte check_sum = 0;
            uint addr = 0;
            Array.Clear(TxBuffer_Frame, 0, 32);
            TxBuffer_Frame[addr++] = 0x9c;
            TxBuffer_Frame[addr++] = 0x5f;
            TxBuffer_Frame[addr++] = 0x91;
            TxBuffer_Frame[addr++] = (byte)(updown_ctrl % 256);
            TxBuffer_Frame[addr++] = (byte)(updown_ctrl >> 8);
            for (int i = 0; i < 31; i++)
            {
                check_sum = (byte)(check_sum + TxBuffer_Frame[i]);
            }
            TxBuffer_Frame[31] = check_sum;
            serialPort1.Write(TxBuffer_Frame, 0, 32);
        }
        catch (Exception err)
        {
                MessageBox.Show("发送失败: " + err.Message, "串口发送提示",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
```

所有的按键发送指令函数都是以上述函数为模型的，所以仅取一例作为代表。所有的发送函数的第一步都是判断串口是否连上，因为这是发送数据的基本要求，只有串口检测是正确的，连上后才会执行下面的步骤，不然就提示“请先打开串口”然后返回。在串口正确连接的情况下，程序会执行 try()函数，这个函数是保证在数据写串口的过程中，有任何地方出现错误都能及时抛出错误，被 catch()函数接收后，正确退出，这样有利于提高程序的鲁棒性。程序先将 TxBuffer 数组赋值上帧头，然后将需要赋值的变量赋值到 TxBuffer 数组上。这里需要注意的是，TxBuffer 数组是 char 型数组，所以赋值的时候如果是 Int 类型或其他类型，要注意转换。赋值完成之后，TxBuffer 的末尾会将数组的累加值存起来(不用担心溢出问题，因为作为校验位，如果这边是一个有溢出丢失的值，那个在检测的地方也会是累加有溢出丢失的值，所以还是可以用于判断数据传输的

正确性)，用于校验使用。

2. 飞行测试模块

飞行测试模块的控制按钮有 3 个 radio button 和一个测试按钮。当选择好要进行测试的模式后，单击“测试”按钮。这时测试按钮会给飞行器发送一个测试飞行指令，飞行器会按照事先写好的飞行动作进行飞行。飞行器在进行这个动作飞行过程中，上位机实时地读取它的飞行参数值，并存储到 Excel 表格中。

```
DataSet QuadrotorData = new DataSet();
QuadrotorData = LoadDataFromExcel("QuadrotorData.xlsx");
button18.Invoke(new EventHandler(delegate { button18.Text = "正在采集数据"; }));
for (int i = 0; i < 200; i++)
{
    switch (test_flag)
    {
    case 1: b[i] = height += 1; QuadrotorData.Tables["Sheet1"].Rows[i]["F" + (1)] =
height; break;
    case 2: b[i] = x_pos; QuadrotorData.Tables["Sheet1"].Rows[i]["F" + (1)] = x_pos;
break;
    case 3: b[i] = y_pos; QuadrotorData.Tables["Sheet1"].Rows[i]["F" + (1)] = y_pos;
break;
    }
    Thread.Sleep(50);
}
```

按以上方式存储好之后，将存储在数组 b 里面的 200 个数据进行分析，首先取出最大值和最小值，计算最大峰峰值。然后将存储在数组_max 和_min 里面的波形峰值进行比较和判断，算出相邻波峰与波谷之间的差值，判断差值走向，并且算出波形个数，最后建立以下 4 个模型。

① 正常平稳飞行控制(如图 5.5.9 所示)，若觉得平稳速度较慢，PID 调节可以略微调大 D，但如果 D 超过一个临界值，则飞行器会出现图 5.5.10 至图 5.5.12 所示的一种情形，则需要再调节回来，找到一个最大 D，则是最合适的 D。如果平稳速度较快，也可不用再调节。

② 此时系统渐变到不稳定(如图 5.5.10 所示)，说明 K_P 参数过大，或者 K_D 参数过小或者过大。建议先调节 K_D 参数，若出现系统稳定性高于本次，则继续调往同一方向 K_D 参数，若无明显效果，则向反方向调整参数。

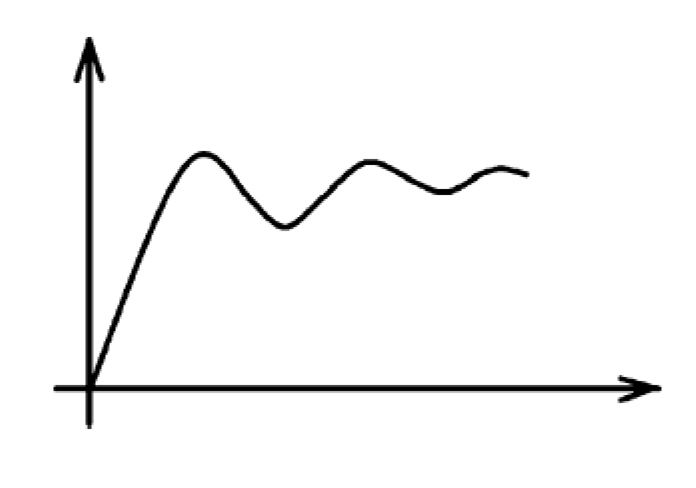


图 5.5.9　正常平稳飞行控制

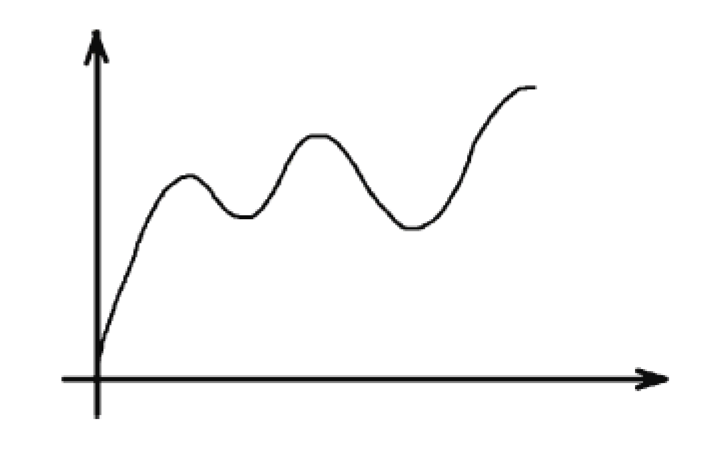


图 5.5.10　系统渐变到不稳定

③ 此时系统直接处于不稳定状态(如图 5.5.11 所示),说明 K_P 参数过大,导致过调,应该减小 K_P 参数,再进行观察。

④ 系统开始高频率震荡(如图 5.5.12 所示),说明 K_D 参数过大,虽然 K_D 参数具有预测误差变化趋势,减小超调量,克服振荡,使系统的稳定性提高,还能加快系统的动态响应速度,减小调整时间的作用,但是过大会导致系统高频率振荡,此时调小 K_D 再次观察。

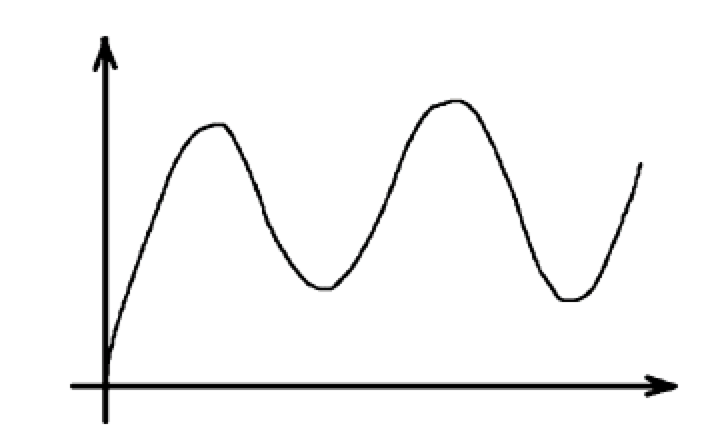


图 5.5.11　系统直接处于不稳定状态

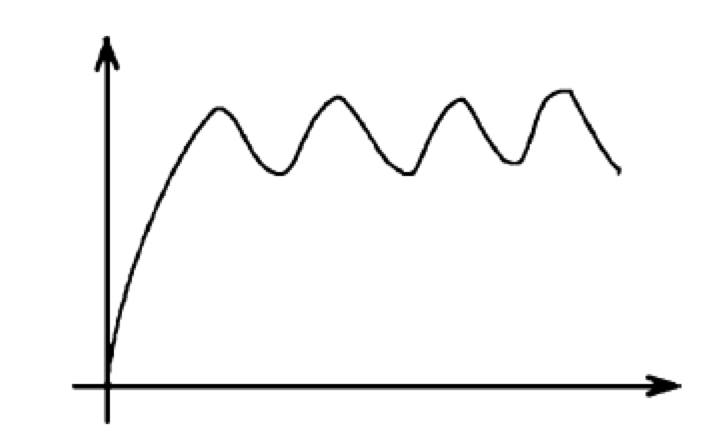


图 5.5.12　系统开始高频率振荡

建立 4 个模型之后,能够通过数据和波形的拟合,大体判断出 PID 参数调节的方向,给调试者一个提示。但过于精细的调节还是需要实际的观察来定的。所以在上位机制作的时候,当数据处理完成存储好了之后,会弹出一个窗口,这个窗口会把处理的数据模型拟合之后给出调节建议,同时把处理的数据原封不动地绘制成波形图,有利于调试者直观地看到飞行器测试飞行的数据。下面给出两幅实际采样图片,该图已经是较为成熟的 PID 参数(如图 5.5.13 所示)。

3. 飞行控制状态模块

这个模块主要用于显示飞行器数据和状态、遥控器的遥控数据等作用,程序流程框图如图 5.5.14 所示。下面先介绍控件。

1) pictureBox 控件

用于绘制仪表盘,并根据数据的刷新,刷新指针,实时指向对应的飞机参量值,并在

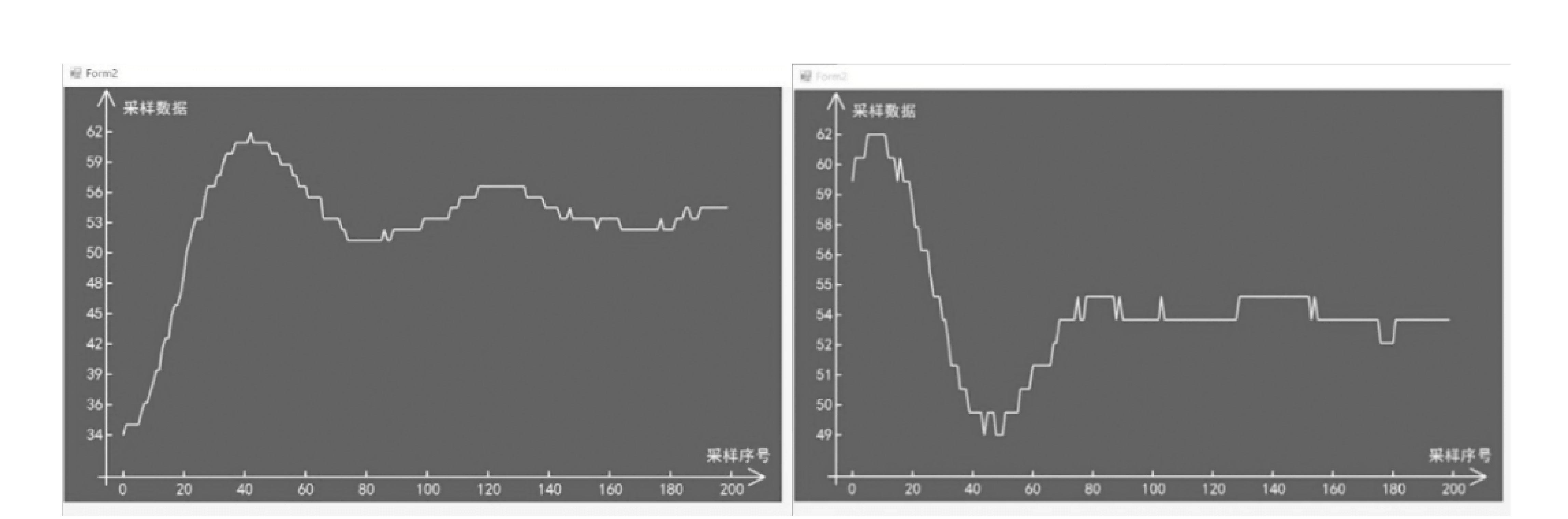


图 5.5.13　PID 参数修改前后的实际采样图片

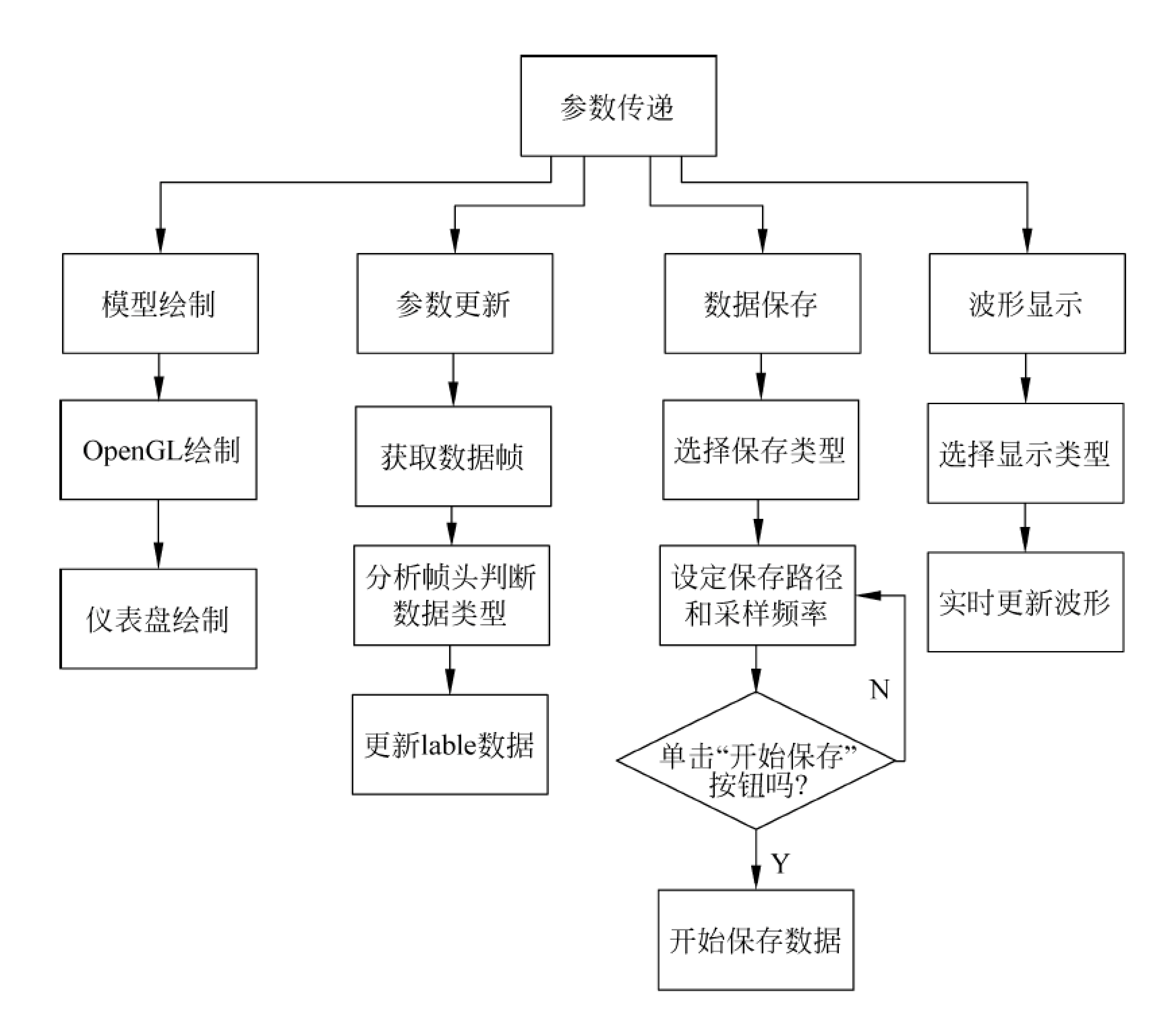


图 5.5.14　飞行控制状态模块流程框图

指针下面显示对应值。

2）OpenGL 控件

用于绘制飞行器模型和三维空间,实时描绘飞行器的状态。

3）progressBar 控件

使用绿色的进度条来显示遥控器的值,能够更加生动、直观地反映出遥控器的控

制量。

4）statusChart 控件

通过单击右侧的 radio button 按钮来选择所要显示的波形数据，当数据开始变化时，控件会实时绘制出选择数据类型的波形，当选择好下面下拉框的保存位置和采样频率后，单击“保存”按钮，即可以保存接下来的 200 个数据点，用于之后的数据处理分析。

数据存储在 Excel 表格中，可由波形显示模块调用查看。

4. 波形显示模块

波形显示流程框图如图 5.5.15 所示。

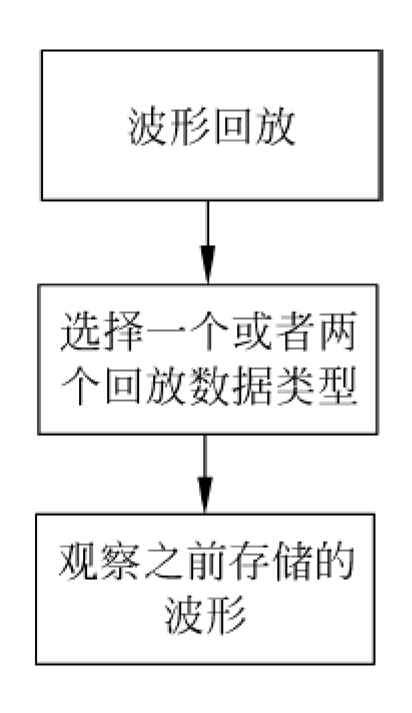


图 5.5.15　波形显示流程框图

此模块的主体是 usercontrol 控件模块，是单独自己定义的一个模块，该模块在通过选择波形类型和数据编号之后，可以调用存储在 Excel 表格里面的数据，绘制成波形，来显示之前存储的数据，并合理地绘画出刻度值。如果勾选两个数据，则会绘画出颜色不同的线条，从而方便比较两次飞行的好坏程度。

第 6 章



四旋翼飞行器制作及调试方法

6.1 飞行控制板 PCB 板制作与调试技术

在 Altium Designer 10 设计环境中原理图设计完成，并经校验无误后便可以转入印制电路板（Printed Circuit Board，PCB）的设计，一般的 PCB 设计流程框图如图 6.1.1 所示。

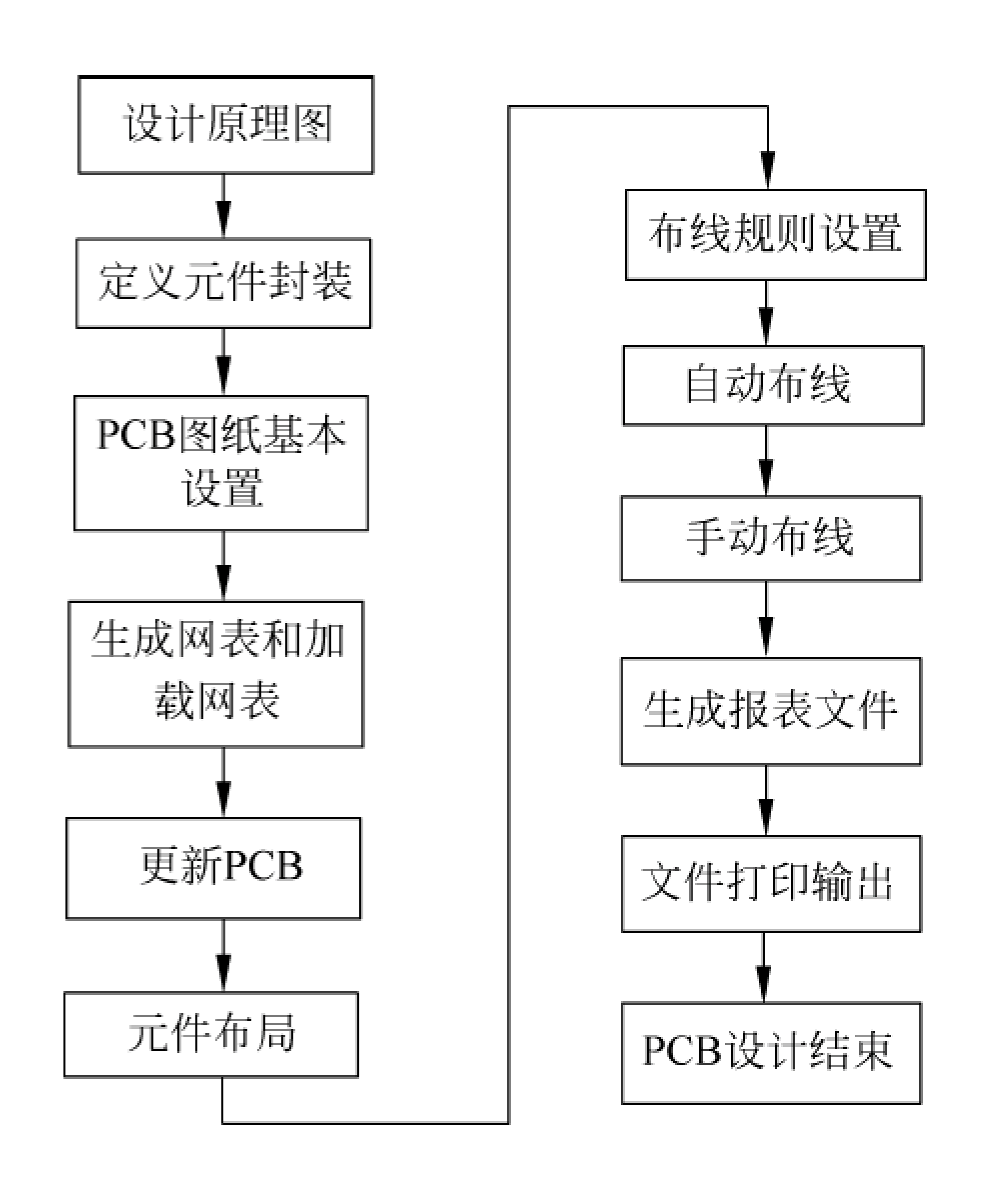


图 6.1.1　PCB 设计的一般流程框图

在将原理图设计转变为 PCB 设计之前，需要创建一个新的 PCB 和至少一个板外形轮廓（Board Outline）。在 Altium Designer 中创建一个新的 PCB 的最简单方法就是运用

PCB 向导,它可让用户根据行业标准选择自己创建的自定义板的大小。在任何阶段,都可以使用后退按钮检查或修改该向导的之前页面。将原理图导入 PCB 前要确保所有与原理图和 PCB 相关的元件库是可用的。因为只有默认安装的集成库被用到,所以封装已经被包括在内。如果工程已经编译并且原理图没有任何错误,则可以使用 Update PCB 命令来产生 ECO(Engineering Change Orders,工程变更命令),它将把原理图的信息导入到目标 PCB 文件。导入原理图生成的网格表之后,便可对元器件进行拖动布局。在板上摆放元器件之前,需要对 PCB 工作环境进行相关设置,如栅格、层以及设计规则。PCB 编辑工作环境允许 PCB 设计在二维及三维模式下表现出来。

在 PCB 设计中,布局是一个重要的环节。布局的好坏将直接影响布线的效果,因此可以认为合理的布局是 PCB 设计成功的第一步。布局的方式分为两种:一种是交互式布局;另一种是自动布局。一般是在自动布局的基础上用交互式布局进行调整。在布局时还要根据布线的情况对元器件、门电路及引脚进行再分配,使其成为便于布线的最佳布局。在布局完成后,还可将设计文件及有关信息进行回注,使得 PCB 中的有关信息与原理图保持一致,以便使今后的建档、更改设计能同步,同时对模拟的有关信息进行更新,使得能对电路的电气性能及功能进行板级验证。

在 PCB 布局过程中,首先要考虑的是 PCB 尺寸大小,PCB 尺寸过大时印制线路长,阻抗增加,抗噪声能力下降,成本也会增加;尺寸过小,则散热不好,且邻近线条易受干扰。其次,在确定 PCB 尺寸后,要确定特殊元器件的位置。最后,根据电路的功能单元,对电路的全部元器件进行布局。

图 6.1.2 所示为布局好的飞行控制板 PCB 图。

考虑到四旋翼飞行器工作的特点,设计应采用小巧、轻便的贴片元器件,这样可以尽量缩小板子的体积以满足飞行器对重量的要求,合理安排各方面的接口电路以便与机架进行组装和后续的调试。根据规则手动布局好元器件位置。

特别应当注意的一点是,由于飞行控制板采用了 6 轴传感器 MPU6050 模块,来获取角度数据,通过 PID 和姿态解算进行飞行,因此对该模块放置的位置有严格的要求。放置角度应使该飞行控制板所对应的四旋翼飞行器飞行模式属于 X 形,应尽量保证 MPU6050 模块位于 PCB 的中央,且在飞行器静止时处于水平状态。

Altium Designer 10 提供了详尽的 10 种不同的设计规则,这些设计规则包括导线放置、导线布线方法、元件放置、布线规则、元件移动和信号完整性等规则。根据这些规则,系统进行自动布局和自动布线。其中重要的规则如走线宽度 Width 要根据信号线承载

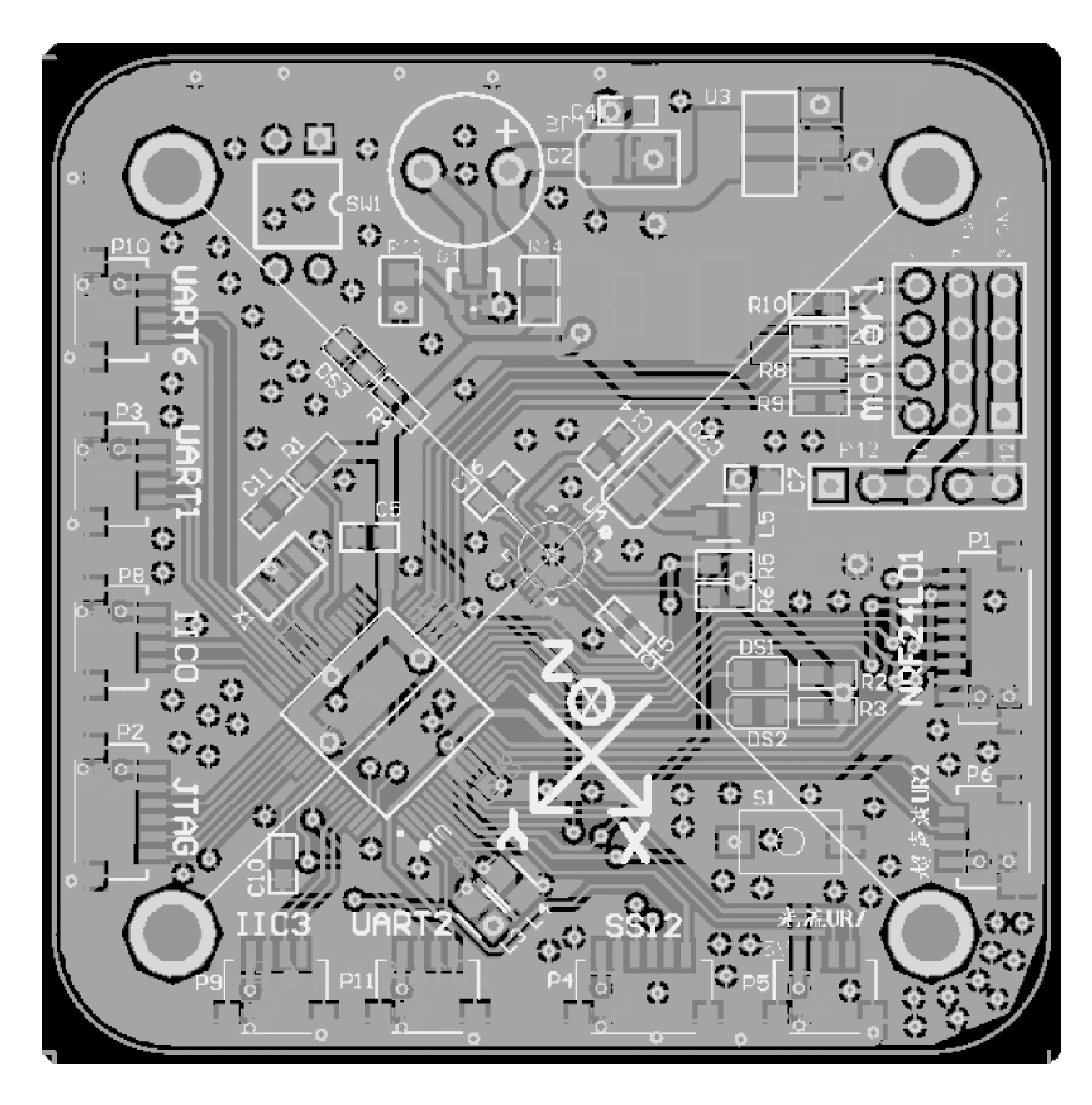
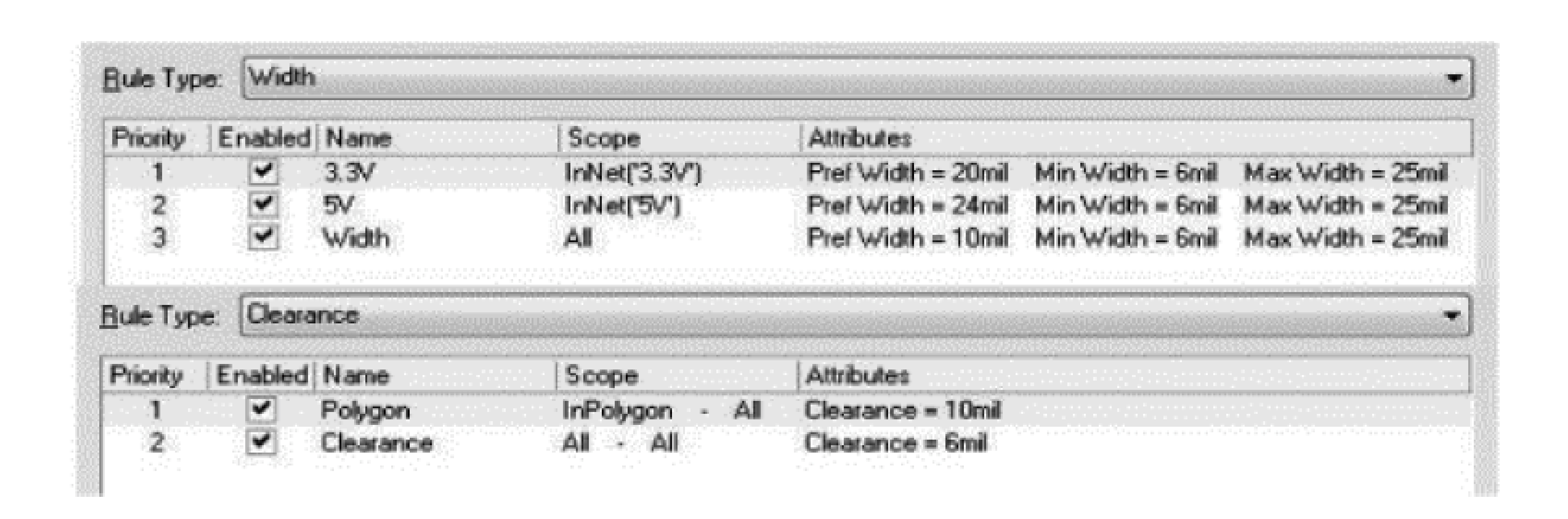


图 6.1.2　布局好的飞行控制板 PCB

的电流大小进行合理设置，走线的间隔 Clearance 距离影响到信号间的稳定，也要对其进行合理的设置。由于本次设计是双面板，很多规则可以按照系统默认的设置。设置好的关键规则如图 6.1.3 所示。



Rule Type: Width

Priority	Enabled	Name	Scope	Attributes
1	✔	3.3V	InNet('3.3V')	Pref Width = 20mil Min Width = 6mil Max Width = 25mil
2	✔	5V	InNet('5V')	Pref Width = 24mil Min Width = 6mil Max Width = 25mil
3	✔	Width	All	Pref Width = 10mil Min Width = 6mil Max Width = 25mil

Rule Type: Clearance

Priority	Enabled	Name	Scope	Attributes
1	✔	Polygon	InPolygon - All	Clearance = 10mil
2	✔	Clearance	All - All	Clearance = 6mil

图 6.1.3　关键设计规则设置

Altium Designer 设计环境的自动布线是靠前面设置好的规则来进行的，因此布局完成和设置好设计规则后便可以进行自动布线，自动布线能最大限度地提高布通率，对于非高频板可以采用先自动布线后手动更改优化的方法。执行 Auto Route→All 命令，在弹出的对话框中定义板子的走线方向和布线策略。本设计是双层 PCB，定义走线方向顶层为水平走向，底层为垂直走向，这样设计可以减少层间的电磁干扰，提高电磁兼容性，布线策略采用系统默认的双层板布线策略，然后单击 Route All 开始自动布线。自动布线完成后可以针对不理想的地方进行手动更改优化，直至 100%布通为止。

布线完成后就可以进一步完善板子的设计，如补泪滴和覆铜。泪滴即在导线和焊盘或导孔之间的一段过渡，过渡的地方呈泪滴状，其可以保护焊盘，避免在导线与焊盘的接触点处出现应力集中而断裂。通过 Tools 菜单中的 Tearsdrops 命令进行补泪滴的操作，如图 6.1.4 所示。

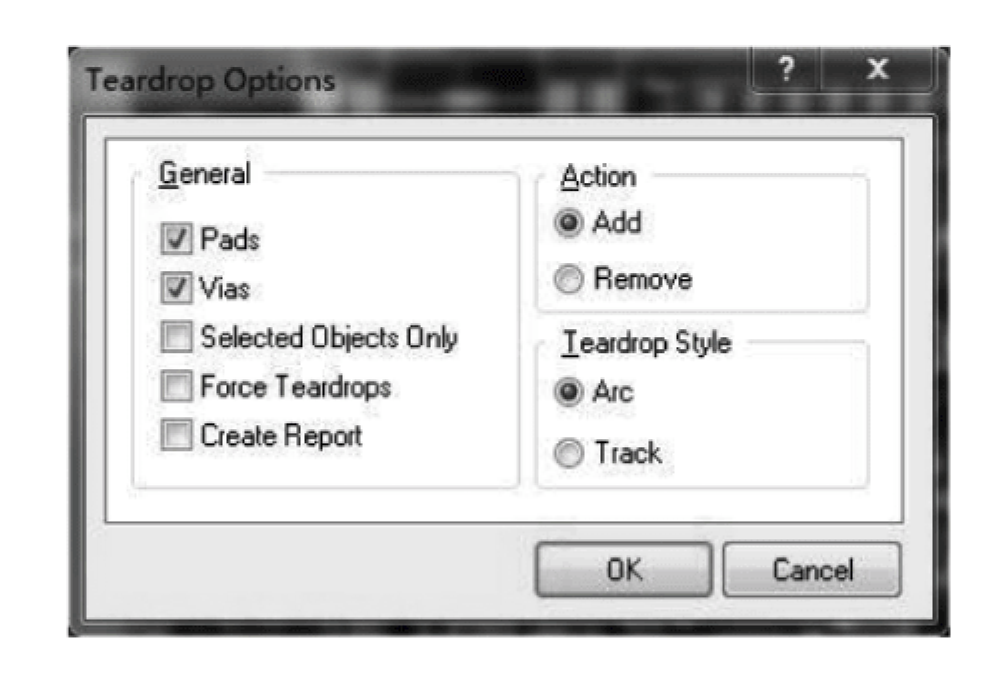


图 6.1.4　补泪滴操作对话框

覆铜就是将 PCB 上闲置的空间作为基准面，然后用固体铜填充，这些铜区又称为灌铜。敷铜的意义在于：减小地线阻抗，提高抗干扰能力；降低压降，提高电源效率；与地线相连，减小环路面积。执行 Place Polygon Pour 命令，打开覆铜设置对话框进行设置，确定后就可以进行覆铜操作了，顶层和底层均要覆铜。

由于设计的 PCB 用的是表贴片元器件封装，同传统的封装相比，它可以减少电路板的面积，易于大批量加工，布线密度高。焊接时一般遵循以下顺序：看体积，先小后大；看板型，先里后外；看规格，先表贴片元器件后直插式元器件。

焊接完成后，就要进行测试。在上电之前先用万用表的欧姆挡测量一下板子上的电源接口是否有短路的情况，无短路后才能上电测试，用 J-Link 硬件仿真器通过 JTAG 口与板子相连，运行 J-LinkCommander 测试飞行控制板是否能被计算机识别，然后在 Keil 开发环境下编写一个简单的测试程序。

6.2　四旋翼飞行器的组装

6.2.1　飞行器的组装步骤

无人机安装所需部件如图 6.2.1 所示，安装步骤如下。

① 将电机导线焊接上香蕉头公头，用于后续与电子调速器连接，如图 6.2.2 所示。

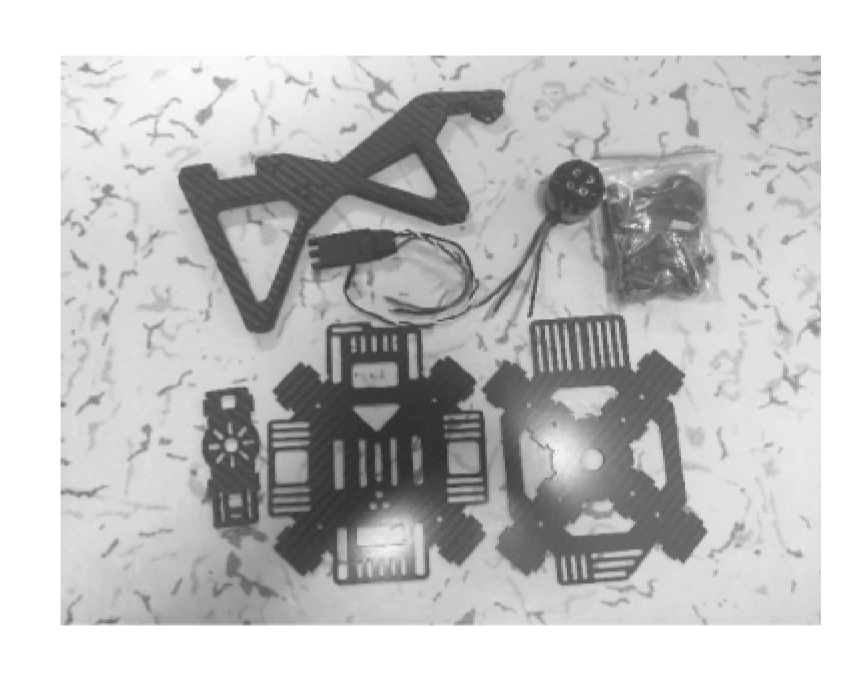


图 6.2.1　无人机安装所需部件

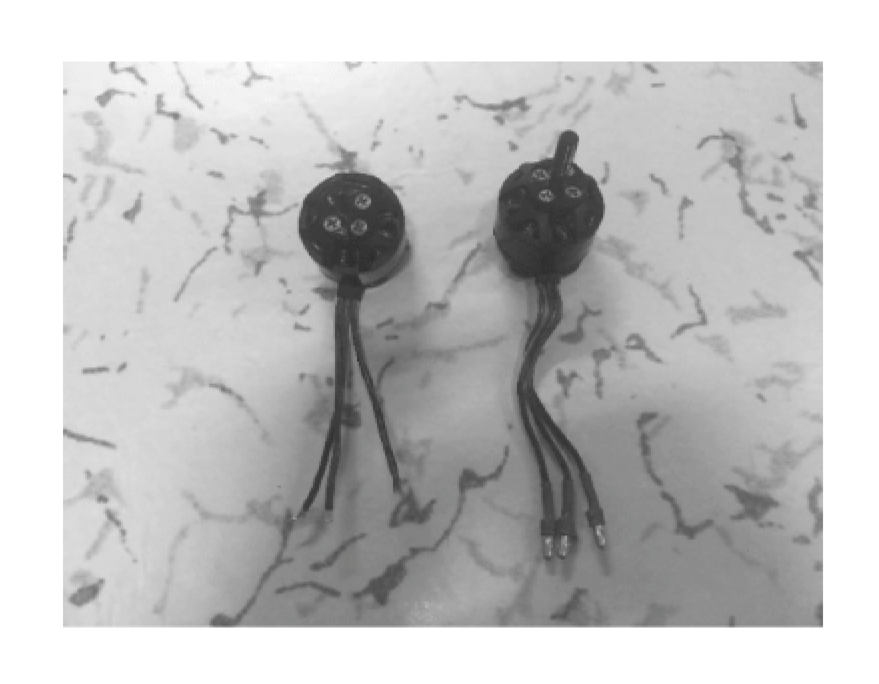


图 6.2.2　电机与香蕉头焊接示意图

② 将电机固定在电机座上，如图 6.2.3 所示。

③ 将电机座与机臂相互连接，如图 6.2.4 所示。

④ 完成机架的安装，注意电机的螺纹方向，顺时针螺纹的电机安装在机架的左后方和右前方，另一组电机安装在剩余两个方向，如图 6.2.5 所示。



图 6.2.3　电机安装在电机座示意图



图 6.2.4　电机与机臂互连示意图

⑤ 连接电机和电调(先随意连接，后续再进行转向调整)，如图 6.2.6 所示。

⑥ 将电调线和电源线焊接在分压板的相应位置，并固定分压板、电源线、电子调速器等，同时从分压板上引出 5V 输出，如图 6.2.7 所示。

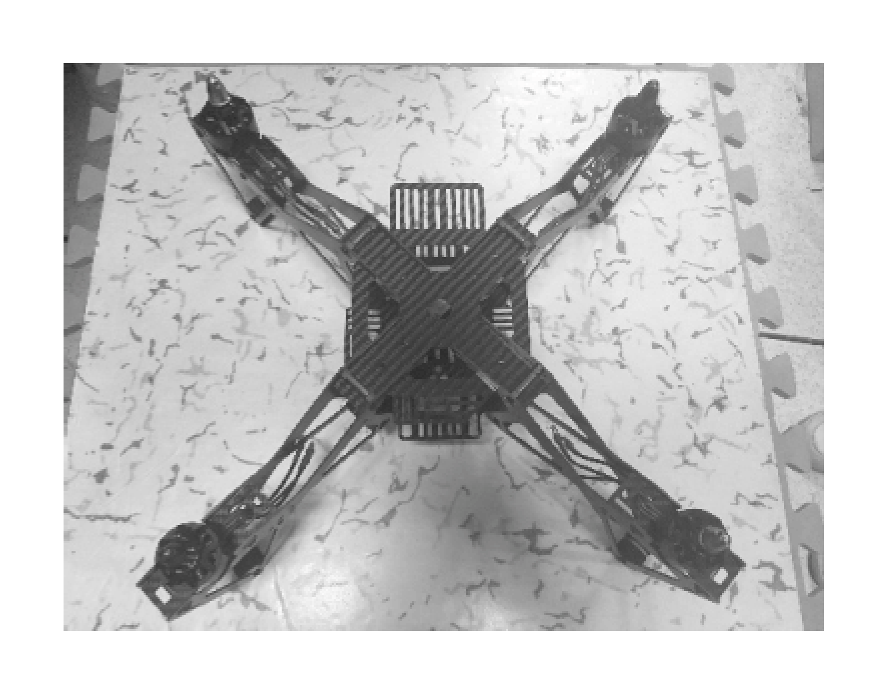


图 6.2.5　机架示意图



图 6.2.6　电调与电机相连示意图

⑦ 根据无人机设定方向，放置和固定飞行控制模块，并将 5V 电源线、电调信号线与飞行控制模块相连，如图 6.2.8 所示。

⑧ 安装各类传感器。装好飞机之后，首先要进行电调初始化，设置各电调的上下限输出，以实现同步，之后就开始调整电机的转向了。首先要了解飞机对角的转向相同，相邻的转向相反，这样才能抵消飞机的自旋。注意调整转向时，一定要拆下桨。从 1 号(X 轴的左边)电机开始作为起点，来判断 2、3、4 号电机方向。接通电源，把遥控油门开到最低，以使转速最低，打开遥控开关，观察顺逆方向，操作 2～3 次验证。若与预期方向相反，则改变 3 根信号线中的任意两根信号线香蕉头的接向。剩下的与电机调试方法相同。

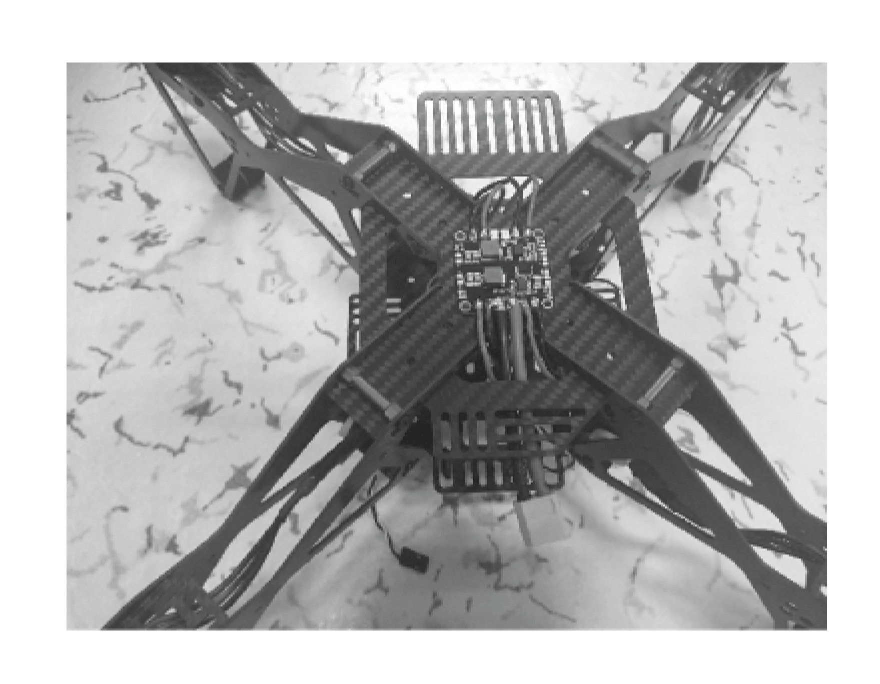


图 6.2.7　电源线、分压板等安装示意图



图 6.2.8　飞行控制模块安装示意图

6.2.2　新组装四旋翼飞行器的第一次调试过程

新组装成的四旋翼飞行器在第一次通电前，先卸下螺旋桨，用万用表确认电源输入正负极两端没有短路情况下接上电池，观察各个模块供电是否正常，无异常则进行以下

步骤。

1. 电调（电子调速器）行程范围设置

电调初始化就是设定电调的最高转速和最低转速，使4个电调实现对PWM波解析的同步，使四旋翼飞机能够实现平衡和稳定。电调一般选择比电机电流大3～4倍为好（如5A电机则选择20A电调），电机转速的控制是单片机输出PWM到电调，PWM的脉冲宽度控制转速，电调与电机连接的3根线的接法控制电机转动的方向，若转动方向不对，3根线中任意两根线的交换连接就会使电机反转。电调行程范围设置的PWM波要求根据型号查看电子调速器使用说明书。校准时电调发出两个“哔”声，表示最大、最小油门捕获，然后就会听到几个“哔”声（每一声代表所使用的电池的一芯），随后一个长“哔”声表示电调已校准。为了方便快速，一般通过编写初始化电调程序对它们行程范围进行设置，运行程序后当听到“嘀嘀，嘀嘀嘀，嘀”声后表明设置成功。

测试电机。在已经校准了电调之后，可以插上电池测试它们，此处还是不要安装螺旋桨。为了便于初学者掌控飞行，将飞行器设置成“Attitude模式”，这样遥控器油门杆控制自旋后松开飞机又能回到原方向；将遥控器设置成自稳模式，这样在用roll与pitch操作控制飞行器的倾斜角度后，松开roll与pitch摇杆时，飞行器将会自动水平。

飞行器放在较水平的地面上，解锁飞行器，给一个小量的油门，所有电机应该以大致相同的速度旋转，并且它们起转也应该在同一时间。如果电机没有在相同的时间起转，旋转也不在同一速度，那么电调仍没有正确校准。当电机工作正常后，前后左右倾斜飞行器时，观察不同位置的电机转速是否能相应做出正确调整。

起飞时的常见问题。电机测试正常后就可以开始起飞了，这时要注意飞机放在起飞位置后才插电，是为了确定飞行器的Attitude正方向位置，起飞时可能会遇到以下问题。

① 刚装的飞行器在起飞时秒翻。

通常是电机顺序错误、电机转动方向错误或者螺旋桨方向（顺时针或者逆时针）安装错误造成的。

② 飞行器在roll或者pitch方向上来回晃动。

通常意味着PID参数值不对，请参阅PID调节部分来正确调整参数。

③ 飞行器在快速下降时来回晃动。

这是飞行器在快速下降时螺旋桨的惯性转动所导致，增加roll或pitch的P值可能会有所改善。

④ 飞行器在起飞时向左或者向右自旋一定角度。

这可能是因为某些电机没有垂直安装或者电调没有校准成功。

⑤ 飞行器总是在无风的环境中朝着某一个方向漂移。

如果只是起飞时偏，飞至空中后姿态能自动调整稳定，那是由于在无人机姿态控制中 I 值为 0 时，飞行器重心偏移所致。如果飞行器在空中总是朝着某一个方向飘，就用遥控器上加减微调值或在飞行控制控制姿态程序中在相应姿态方向上增加固定偏移量。要特别强调的是，因为电池较重，它的安装位置是决定飞机重心的关键因素，所以一定要根据飞机重心选择最佳位置并且每次安装都一致。

⑥ 飞行器在空中无法完美地保持高度位置。

在没有测高传感器的情况下，在自稳模式下想要定高定点，必须不断用遥控器修正飞行器姿态。在装有测高传感器时，一般是测高数据没有滤波或 PID 参数没有调整好。

2. 遥控器参数设置

遥控器采用 9 通道 2.4GHz 接收机，4096 分辨率（PCMS 4096 制式下），在遥控飞机前需要设置控制值范围。

① 正反设置。调遥控器操纵杆的控制方向的正反。

② 舵角设置。调上下限。下限 674，上限 1467。

③ 辅助设置。调中值，1071。

④ 遥控器关 674，遥控器关开 1467，另外还有表 6.2.1 中 4 个通道的值。

表 6.2.1 遥控器通道配置

通道	最小	中值	最大
副翼（左右）	634	1072	1434
升降（前后）	590	1072	1411
油门（高低）	589	1049	1398
方向（偏航）	634	1072	1434

其他在调试时总结的经验如下。

① 在飞行控制程序中添加当飞机撞物致使飞机倾斜大于 45°时让桨停转；人为遥控飞机或在飞行控制程序高度 PID 尚未调试完成前当飞机达到危险高度时立即自动运行紧急降落程序。

② 遥控器与飞机一旦飞出遥控范围或故障导致通信中断，飞行控制程序中应加入立刻启动自保护的飞机紧急降落程序。

③ 超声波安装时，如果用立柱螺钉安装在机架上，起飞后机架的抖动会使超声波跟随着机械装置发生振动，使得高度数据不稳。应该用海绵包裹超声波，然后用双面胶将其粘在机架上，起到避振的效果，这样就可以避免超声波在静态时测量正常，而在起飞后高度数据乱跳的现象。

④ 如果有几组人员都在相邻的场地上调试飞机，遥控无线通信用的都是 nRF 模块，为了使各个组之间调试不致相互影响，nRF 模块初始化时应将通道号错开。

6.3　常见调试问题与解决方案

6.3.1　PID 参数调整经验

PID 参数调节最直观的方法就是通过波形图来观察调节的规律。本系统在进行高度 PID 和位移 PID 参数调节时，将调节对象的实时数据发送回来，在上位机中显示其变化的波形，通过观察波形变化的规律来细调 P、I、D 这 3 个参数。具体调节时可以参照下面的 PID 整定口诀：

参数整定找最佳，从小到大顺序查。
先是比例后积分，最后再把微分加。
曲线振荡很频繁，比例度盘要放大。
曲线漂浮绕大弯，比例度盘往小扳。
曲线偏离回复慢，积分时间往下降。
曲线波动周期长，积分时间再加长。
曲线振荡频率快，先把微分降下来。
动差大来波动慢，微分时间应加长。
理想曲线两个波，前高后低四比一。
一看二调多分析，调节质量不会低。

最后整定好 PID 参数后，得到的波形图应该类似图 6.3.1。图 6.3.1 是飞行器高度控制 PID 参数调好后的实时高度数据波形图，最上面的那条波形是飞行器的油门值，中间部分是飞行器的实时高度值，最下面是高度控制 PID 调节器的输出值，PID 调节器中输入当前高度和目标高度的误差，输出调节值，作用在飞行器的油门上，最终使飞行器的高度保持在目标高度附近上下抖动，实现飞行器的自动定高功能。

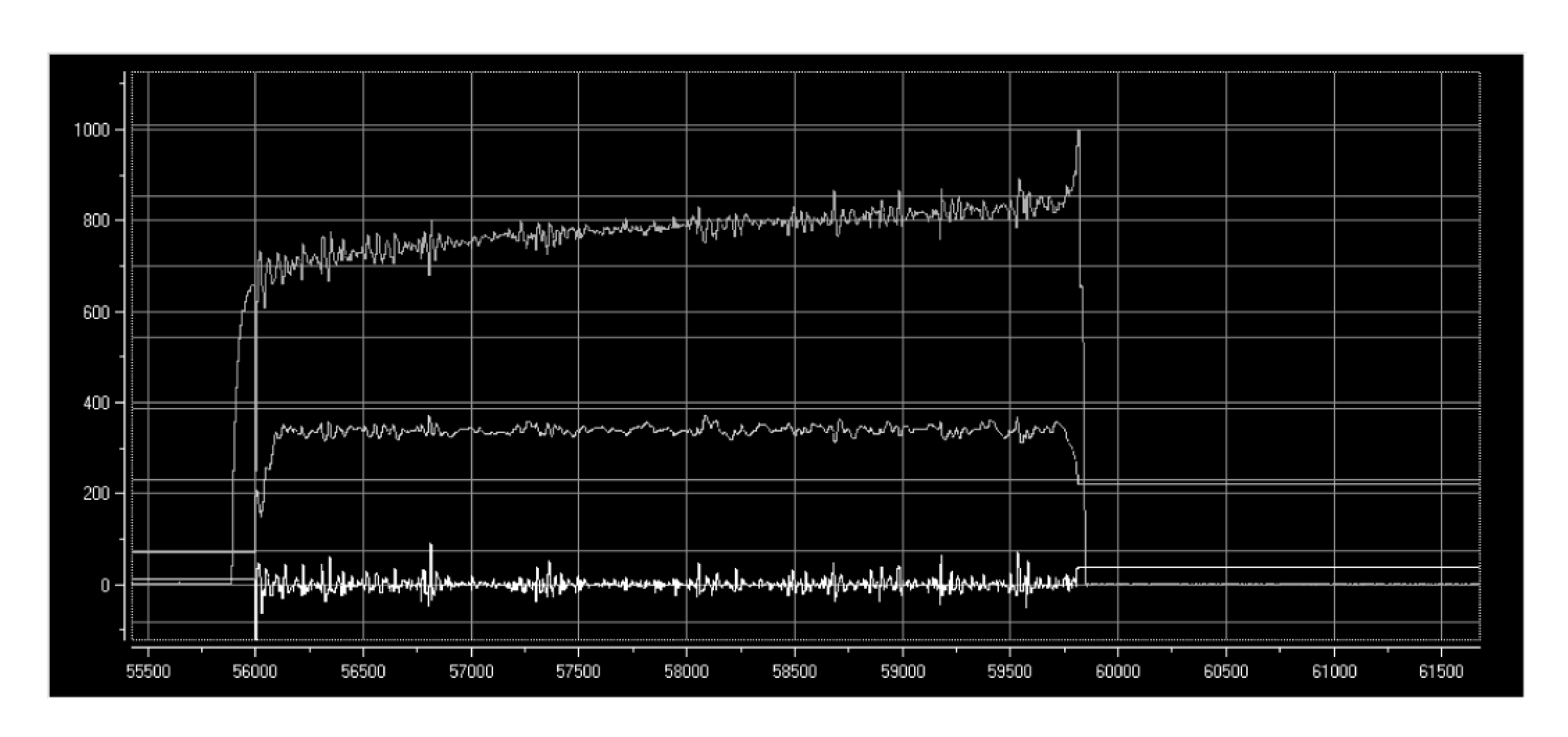


图 6.3.1 高度调节波形

此外，也可以使用凑试法进行 PID 调节，它的优点是要求较低、操作简单易行。测试时，对参数实行先比例、后积分、再微分的整定步骤，具体如下。

① 先调节比例部分。确定比例系数 K_P时，首先去掉 PID 的积分项和微分项（即令 $K_I=0$、$K_D=0$），使之成为纯比例调节。输入设定为系统允许输出最大值的 60%～70%，比例系数 K_P由 0 开始逐渐增大，直至系统出现振荡；再反过来，从此时的比例系数 K_P逐渐减小，直至系统振荡消失，且系统反应快。记录此时的比例系数 K_P，设定 PID 的比例系数 K_P为当前值的 60%～70%。

② 加入积分控制。如果在比例调节的基础上系统的静差不能满足设计要求，则需加入积分环节。调节时首先置积分系数 K_I为一较大值，并将经第一步得到的比例系数略微缩小（如缩小为原值的 0.8 倍），然后减小积分时间，使在保持系统良好动态性能的情况下，静态误差得到消除。

③ 加入微分控制。若使用比例积分调节器消除了静态误差，但动态过程经反复调整仍不能满意，则可加入微分环节，构成比例积分微分调节器。在调整时，可先置微分系数 K_D为 0。在第(2)步整定的基础上，增大 K_D，同时相应地改变比例系数和积分系数，逐步凑试，以获得满意的调节效果和控制参数。

了解了以上两种 PID 调试方法后，解释一下 PID 参数调整的依据。

① 在偏差比较大时，为使尽快消除偏差，提高响应速度，同时为了避免系统响应出现超调，K_P取大值，K_I取零；在偏差比较小时，为继续减小偏差，并防止超调过大、产生振

荡、稳定性变差，K_P值要减小，K_I取小值；在偏差很小时，为消除静差，克服超调，使系统尽快稳定，K_P值继续减小，K_I值不变或稍取大值。

② 当偏差与偏差变化率同号时，被控量是朝偏离既定值方向变化，而当偏差与偏差变化率异号时，被控量朝既定值方向变化。当被控量接近定值时，异号的微分作用阻碍积分作用，避免积分超调及随之而来的振荡，有利于控制；而当被控量远未接近各定值并向定值变化时，则由于比例作用和微分作用反向，将会减慢控制过程。在偏差比较大时，偏差变化率与偏差异号时，K_D值取零或负值，以加快控制的动态过程。

③ 偏差变化率的大小表明偏差变化的速率，$e(t)-e(t-1)$越大，K_D取值越小，K_I取值越大；反之亦然。同时，要结合偏差大小来考虑。

下面介绍一种 PID 参数半自动整定方法。

在本应用中采用的 PID 参数整定方法为临界比例度法，比例度 $\delta=1/K_P$，积分时间 $T_I=\mathrm{d}t/K_I$，微分时间 $T_I=\mathrm{d}t\cdot K_D$，其中 $\mathrm{d}t$ 为 PID 调节周期。

根据临界比例度整定法，将 K_I和 K_D置零后选择合适的 δ，使无人机在调整方向上出现等幅振荡，此时 δ 记为δ_k，记此时无人机的振荡周期为 T_k，则按经验整定出的 δ、T_I、T_D值的关系如表 6.3.1 所示。

表 6.3.1　临界比例度整定算法

δ	T_I/min	T_D/min
$1.6\delta_k$	$0.5T_k$	$0.25T_I$

完成该算法的核心代码如下：

```
Tk = timeused / (600.0 * n);
Ti = 0.5 * Tk;
Td = 0.25 * Ti;
switch (PID_radiogroup.getCheckedRadioButtonId())
{
case R.id.rosx_radiobtn:
    deltak = 1.6 / (Double.valueOf(posx_P_txt.getText().toString()));
    break;
case R.id.posy_radiobtn:
    deltak = 1.6 / (Double.valueOf(posy_P_txt.getText().toString()));
    break;
default:
    break;
}
```

```
Kp = 1/deltak;
Ki = 0.025 * 1/Ti;
Kd = Td/0.025;
```

另外，还需要注意的是无人机的 PID 调节同温度调节阀门调节等 PID 调节有很大的不同，无人机的 PID 调节不仅要让输出结果恒定在一个值附近，输出值还不能拥有高频抖动，如果存在高频抖动，无人机很可能就会失控。因此在无人机的 PID 调试中，数据的平滑变动是最为重要的；否则就有可能不可避免地增加输出的波动范围。

6.3.2 减少机械振动以提高飞行稳定性

四旋翼飞行器 4 个轴的桨翼应处于同一高度平面，轴的中心处安放飞行控制部分，以使得整个四轴重心稳定。轴翼应固定安放在机身的 4 个轴的末端。

在 PCB 设计时，钻孔应设计得稍微大一些，便于安放避振垫片。由于飞行控制板上所安放的传感器都有很高的精度，因此机身的振动会对它们的读数造成误差。安放垫片可以分散机身与其他部分接触的压强，起到避振的作用，减小了对传感器读数的影响。

同时，对于连接在飞行控制板外部的传感器模块，如超声波模块，其实时读数可以判断飞机的当前高度。应在其周边贴上海绵来减少振动带来的高度误差。

第 7 章



成 功 案 例

7.1 竞赛作品 1——2015 年(瑞萨杯)全国大学生电子设计竞赛

7.1.1 题目要求

设计并制作一架带航拍功能的多旋翼自主飞行器。飞行区域俯视图和立体图分别如图 7.1.1 和图 7.1.2 所示。

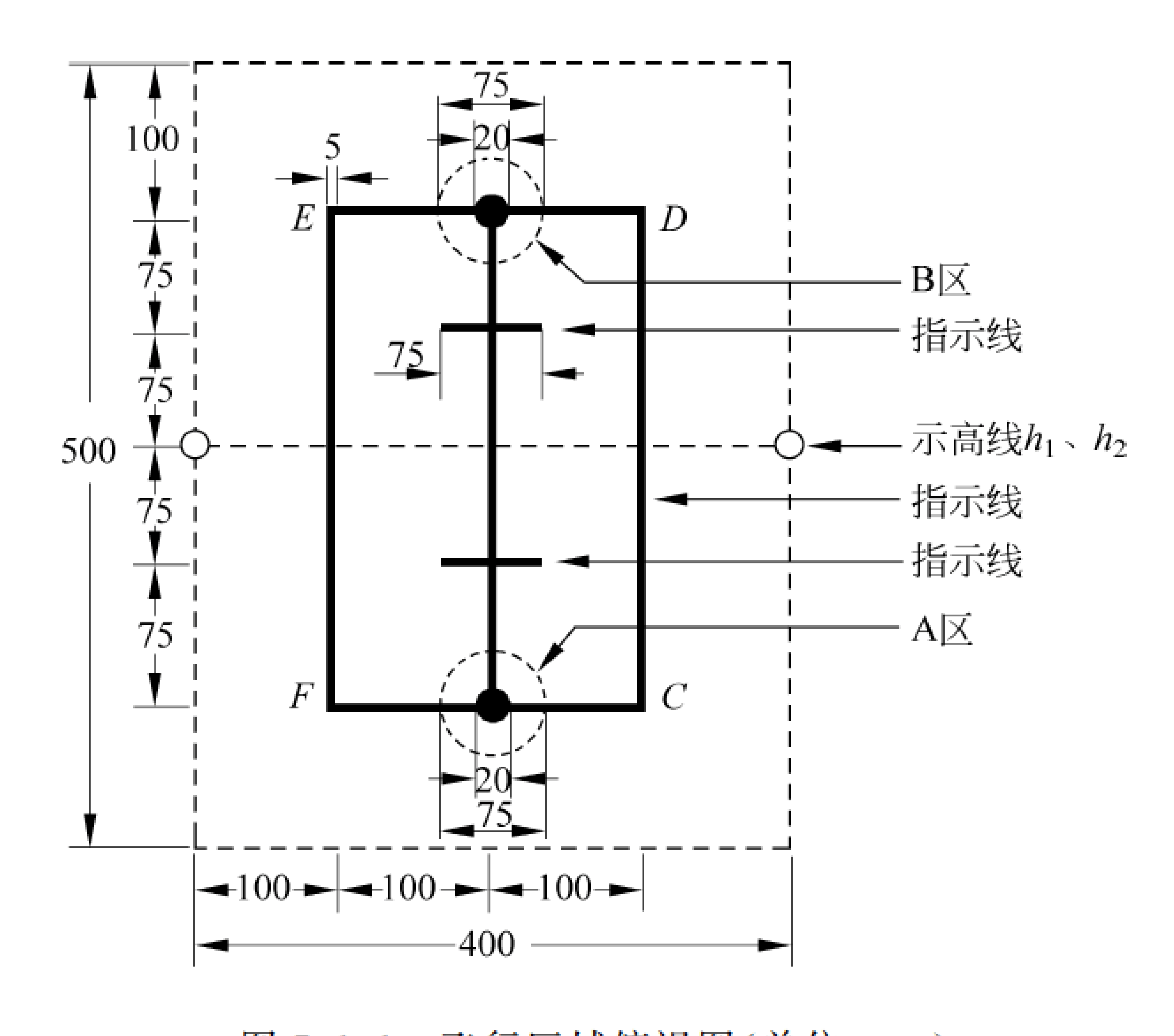


图 7.1.1　飞行区域俯视图(单位：cm)

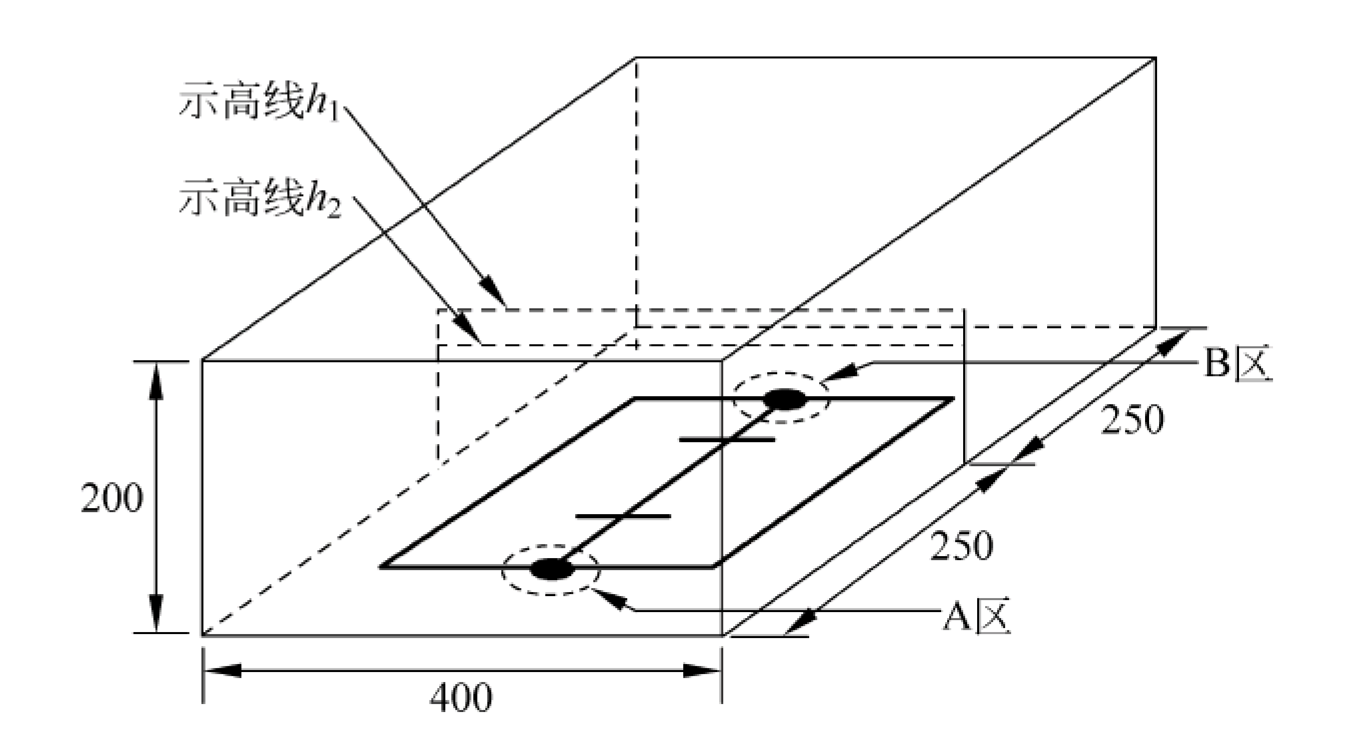


图 7.1.2　飞行区域立体示意图(单位：cm)

1．基本要求

① 多旋翼自主飞行器(下简称飞行器)摆放在图 7.1.1 所示的 A 区，开启航拍，一键式启动，飞行器起飞；飞向 B 区，在 B 区中心降落并停机；航拍数据记录于飞行器自带的存储卡中，飞行结束后可通过 PC 回放。飞行高度不低于 30cm；飞行时间不大于 30s。

② 飞行器摆放在图 7.1.1 所示的 A 区，一键式启动，飞行器起飞；沿矩形 CDEF 逆时针飞行一圈，在 A 区中心降落并停机；飞行高度不低于 30cm；飞行时间不大于 45s。

③ 制作一个简易电子示高装置，产生示高线 h_1、h_2(如激光等)，h_1、h_2位于同一垂直平面，飞行器触碰 h_1、h_2线时该装置可产生声光报警。示高线 h_1、h_2的高度在测试现场可以调整。调整范围为 30～120cm。

2．发挥部分

① 飞行器摆放在 A 区，飞行器下面摆放一小铁板 M1，一键式启动，飞行器拾取小铁板 M1 并起飞。飞行器携带小铁板 M1 从示高线 h_1、h_2间飞向 B 区，并在空中将小铁板 M1 投放到 B 区中心；飞行器从示高线 h_1、h_2间飞回 A 区，在 A 区中心降落并停机。飞行时间不大于 30s。小铁板 M1 形状不限，质量 20g、100g、200g 三挡自选，质量重得分高。h_1、h_2高度差小得分高。

② 飞行器摆放在 A 区，小铁板 M2 摆放在 B 区任意位置；一键式启动，飞行器飞到 B 区寻找并拾取小铁板 M2，携带小铁板 M2 飞回 A 区，在 A 区中心降落并停机。飞行高度不低于 30cm；飞行时间不大于 30s。小铁板 M2 为边长 5cm 的正方形，重量不限。

7.1.2 系统方案

本系统主要添加和修改了控制模块、薄铁片拾取模块、循迹模块。下面分别论证这几个模块的选择。

1. 控制系统的选择

依据本题目的要求，本系统选用组委会提供的瑞萨的R5F100LEA单片机作为主控芯片来控制飞行器的飞行姿态与方向。

2. 循迹模块的选择

一般而言，需要识别地面上的图像信息，需要在无人机上安装摄像头，对摄像头拍摄的图像进行分析和处理。在本题目中，飞机循线飞行只利用到了摄像头一个维度上的数据，考虑到普通摄像头数据量庞大不易处理，采用线性CCD摄像头(1像素×128像素)采集图像，让无人机实现循线飞行。

3. 薄铁片拾取的选择

方案一：在飞行器起飞时由系统控制机械臂拾取起铁片，到达B区放松机械臂，投下薄铁片。缺点：机械臂重量大，对飞行器的飞行姿态影响较大，薄铁片厚度非常小，不易拾取。

方案二：采用电磁铁拾取，用瑞萨MCU控制电磁铁，在飞行器起飞时吸取铁片，到B区后投下铁片。优点：电磁铁体积小而且有较强的拾取能力，操作方便。

综上所述，选择既经济又灵活的电磁铁作为薄铁片的拾取工具，采用方案二。

7.1.3 实现方案

1. 图形识别

在无人机上安装线性CCD摄像头，摄像头沿水平方向放置，将摄像头传回的一维灰度数据进行二值化处理后，根据地面上的黑线占用的所有连续像素点的中间值，判断无人机相对于黑线的左右位置。中间值小于64，则无人机在黑线右侧，若中间值大于64，则无人机在黑线左侧。

利用黑线占用像素点的范围和高度进行拟合，可以计算出黑线的实际宽度，用此宽度值判断飞行起点、终点和标志点，从而实现能在指定的地点让无人机做出对应

动作。

若线性 CCD 的数据从一侧的像素点开始都为黑色，而另一侧为连续白色，则说明无人机处于正方形的直角转角处，用此方法判断无人机何时需要转弯。

2. 循迹飞行控制

① 一键起飞。利用飞行控制板上的按键开关实现。

② 高度控制。采用超声波检测无人机高度，利用 PID 调节控制无人机定高。

③ 直线前进。给无人机俯仰方向一个较小的倾角，或让无人机交替前倾和后倾。通过控制倾角大小或前倾与后倾的时间比例，使无人机以合适的速度匀速前行，使线性 CCD 摄像头能够正确识别地面黑线和标志。

④ 左右调整。根据上文所述，由线性 CCD 黑色像素点的中间值，可以判断无人机与地面黑线的相对位置。根据中间值与所有像素点的中间值 64 的差值，进行 PID 计算，将 PID 的输出值直接由无人机横滚方向控制，即可让无人机保持在黑线上方前进。

⑤ 转弯控制。如上文所述方法，当无人机检测到正方形的弯角时，利用无人机上六轴传感器测量和计算得到精确的偏航角度，直接改变飞行控制程序中偏航方向的控制量，使无人机实现 90°的转向。为了维持无人机转向时的稳定，偏航方向的控制量改变应缓慢循环进行。

⑥ 标志点动作。根据题目要求，无人机要在固定的标志点起飞，降落以及拾取和投放铁片，因此需要根据 CCD 测得的宽度数据，对标志点进行计数，从而实现在规定的标志点实现规定的动作。

3. 薄铁片的丢放和拾取控制

根据前文所述的标志点计数，可以判断无人机当前处于哪个标志点之上。在规定的标志点上，完成丢放和拾取铁片的控制。

在 7.1.1 节中，根据电磁铁的通电具有磁性、断电磁性消失的原理，对于发挥部分①，从 A 起飞时让单片机控制电磁铁通电，让飞行器吸取薄铁片飞向 B 区，到达 B 区后让电磁铁断电，从而投下薄铁片，让其落到 B 区，程序流程图如图 7.1.3 所示。对于发挥部分②则

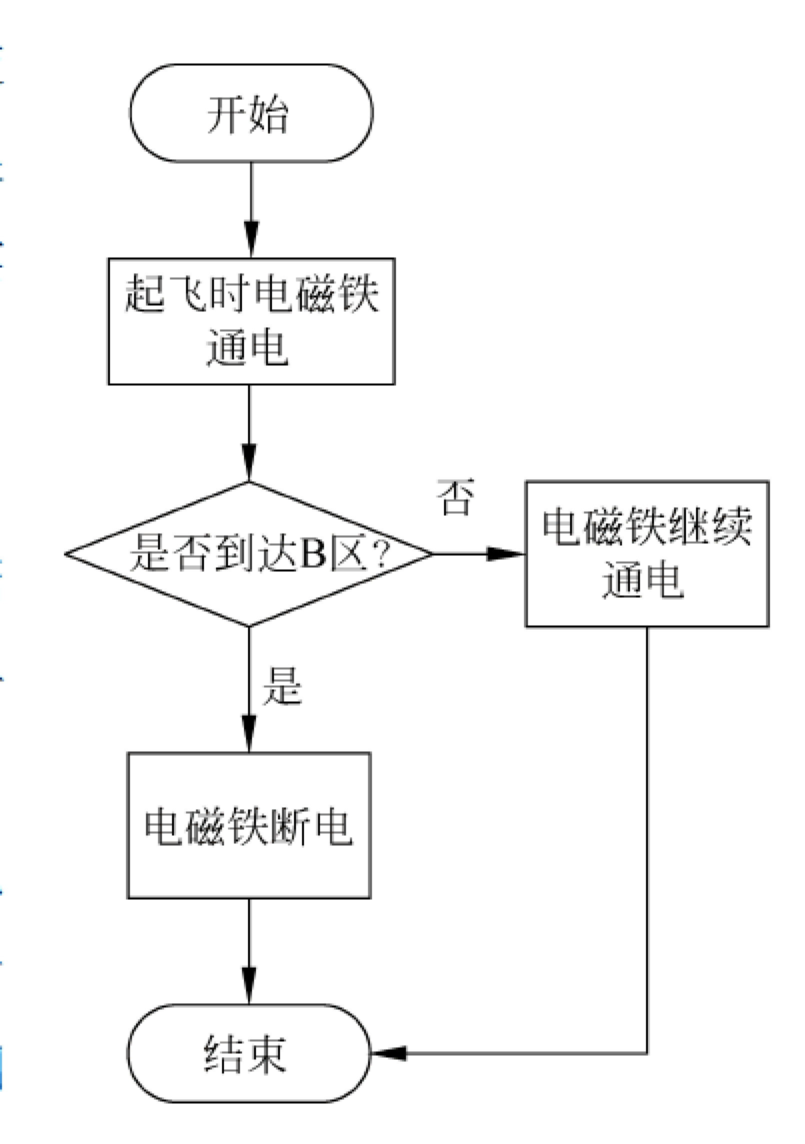


图 7.1.3 拾取铁片流程图

让无人机飞往 B 区后降落，控制电磁铁通电，再让无人机起飞和返航。

4. 模式切换

题目中要求无人机具有多项功能、多次飞行，因此，利用飞行控制板上的拨码开关切换不同的模式，实现不同要求的飞行功能。

7.1.4　软硬件方案

1. 硬件方案

本四旋翼飞行系统由瑞萨飞行导航板、飞行控制板、线性 CCD 模块、超声波模块和电磁铁构成，由瑞萨飞行导航板接收和处理传感器数据，由飞行控制板进行飞行控制和电磁铁控制。各个传感器完成如上文所述功能。硬件框图如图 7.1.4 所示。

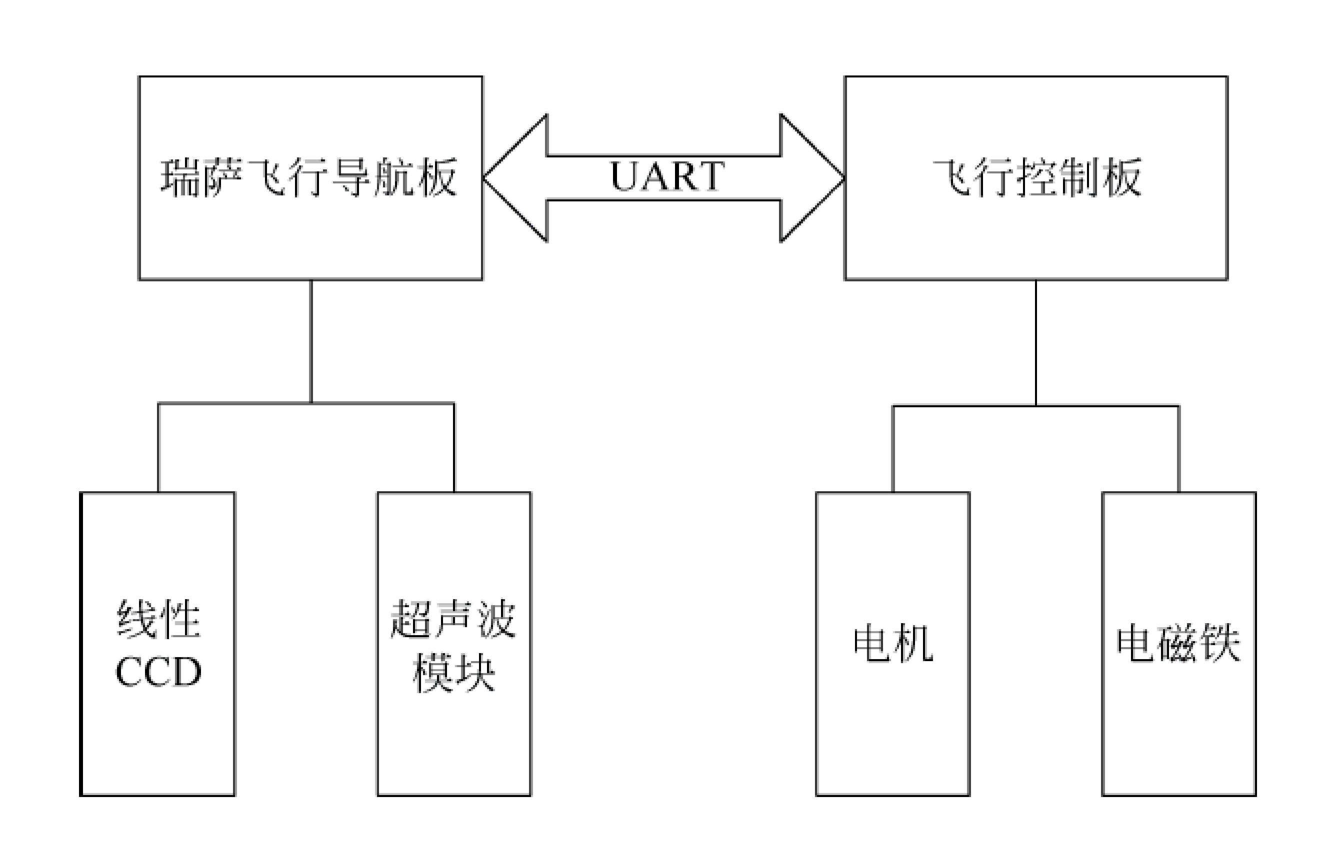


图 7.1.4　硬件框图

2. 软件方案

根据竞赛要求，无人机上必须使用瑞萨芯片作为飞行导航板，而且具有 Tm4C123G 芯片的飞行控制板也必不可少，因此软件上也需要在两个平台上进行编程。瑞萨芯片采用 CubeSuite+软件进行程序的编写，用 Renesas Flash Programmer V2.01 软件将编写好的程序烧写入瑞萨芯片。飞行控制板则使用 IAR Workbench 进行编写和烧录。

瑞萨芯片中的程序实现了传感器数据的采集和数据的预处理，如线性 CCD 数据的处理以及计算地面黑线的位置、宽度、转角等，通过串口将数据传输到飞行控制板。飞行控制板中的程序则主要实现循线飞行的控制，投放和拾取铁片的控制，以及实现一键起

飞和不同模式的切换等。瑞萨芯片程序流程框图如图 7.1.5 所示，TI 芯片程序流程框图如图 7.1.6 所示。

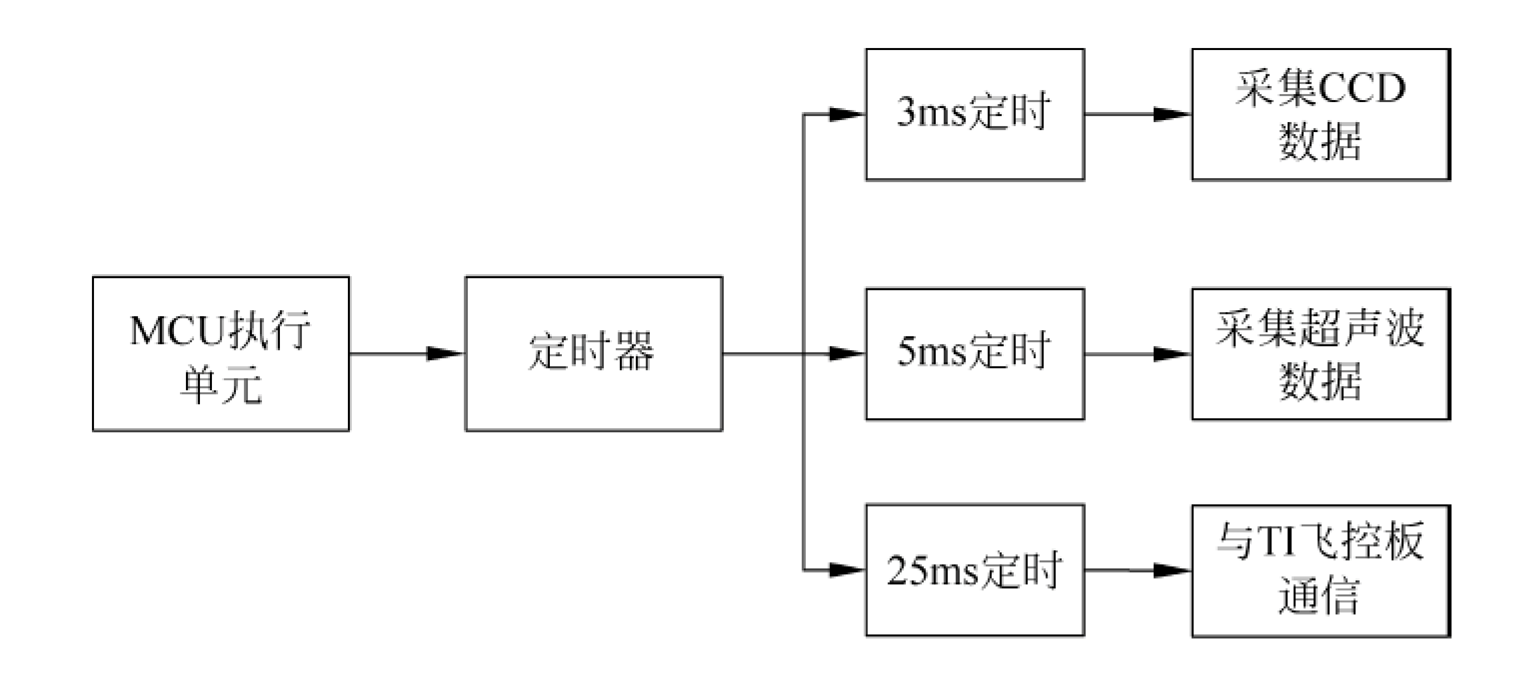


图 7.1.5 瑞萨芯片程序流程框图

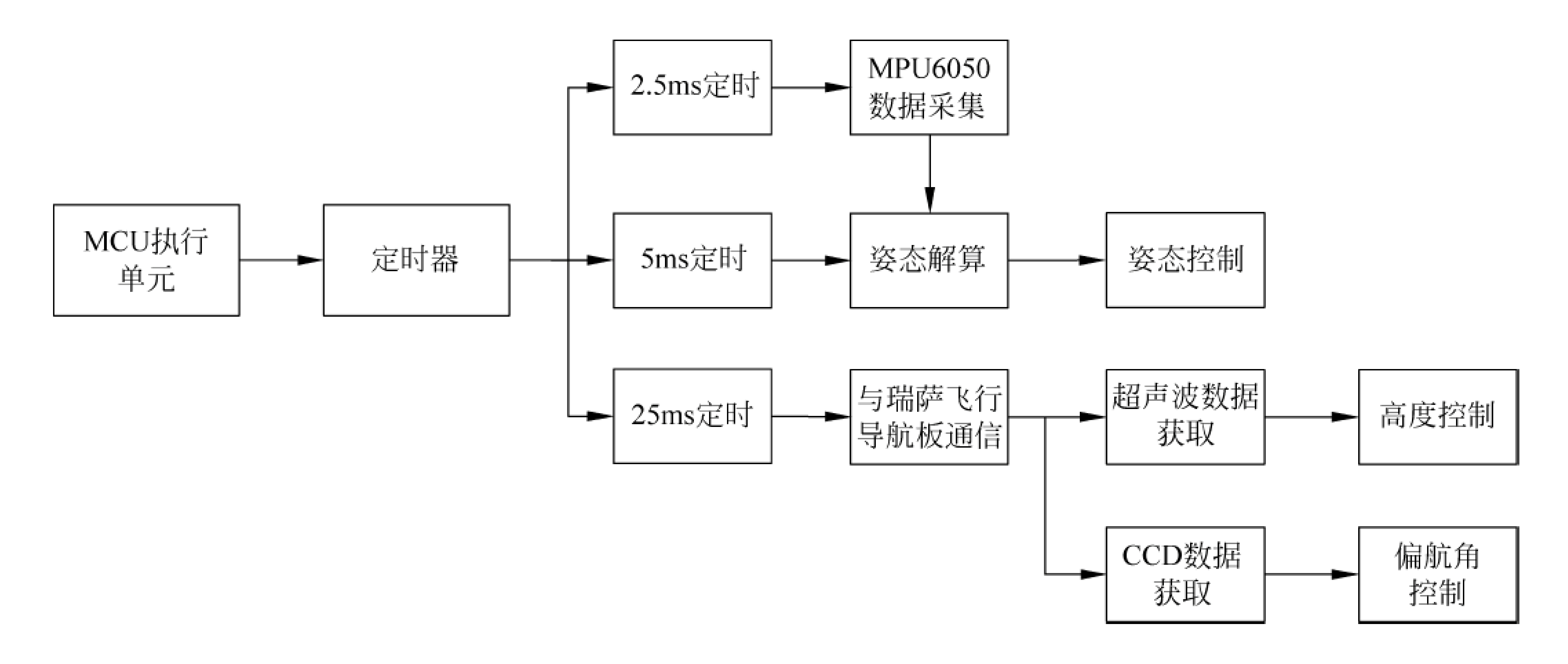


图 7.1.6 TI 芯片程序流程框图

7.1.5 测试

1. 测试方案

将飞行器放在圆形区域 A 或 B，让单片机自主控制飞行器飞行，测试过程中，逐个功能依次进行测试，观察飞行器的飞行高度与飞行方向和时间，若飞行器不能按预定的方案飞行，则需要对无人机的程序或者参数进行调整再次测试。

2. 测试条件

在如题所示场地中进行测试。无人机在水平状态下，通过按钮一键起飞。

3. 测试结果

经测试，无人机能够完成直线飞行，绕正方形飞行，拾取和放置小铁板，为了稳定起见，绕正方形飞行时间较长。另外，由于电磁铁吸引范围不大，拾取电磁铁对铁板摆放位置有较高的要求，拾取放置铁片的方案能够实现基本功能。

4. 结果分析

由于无人机是纯自主飞行，因此当多次飞行电量降低后，无人机容易出现无法完成预定飞行目标，甚至无法稳定飞行的情况。在没有电量自适应调整的前提下，应尽量保证采用有充足电量的电池进行飞行测试，并尽量保证飞行时间较短。

7.2 竞赛作品2——2016年(TI杯)上海市大学生电子设计竞赛

7.2.1 题目要求

1. 设备要求

旋翼无人机(轴距小于600mm)，自带处理平台、摄像头(视角小于90°)，有无线遥控装置可以控制起飞。

2. 测试环境

室内场地(10m×10m，高度10m)，其中随机放置4个不同形状的标志(黑色)，包括大正方形(50cm×50cm)、圆形(直径40cm)、正三角形(边长40cm)、小正方形(30cm×30cm)，如图7.2.1所示。

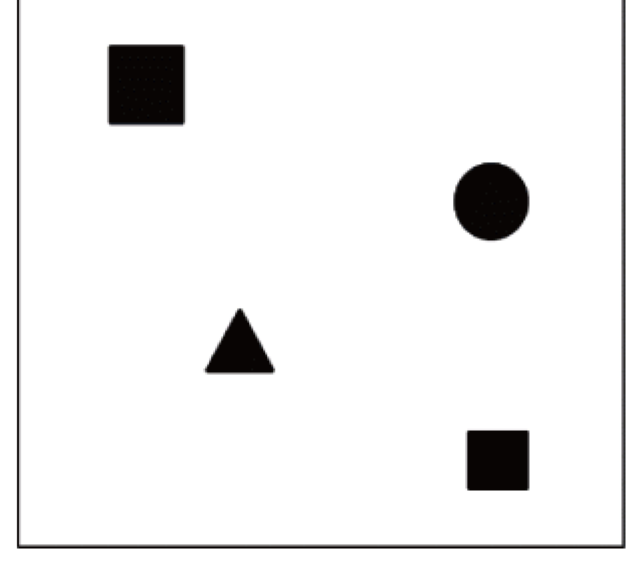


图7.2.1　场地示意图

3. 完成操作

无线遥控装置发出起飞命令，飞机从大正方形中心起飞，自动按照圆形、三角形、小正方形顺序飞行，在每一个标志物上完成降落和起飞，最后降落到小正方形中心。无线遥控装置不能干预飞行过程，仅用于异常情况时紧急降落。

7.2.2 系统方案

本系统主要添加和修改了高度测量模块、图像识别模块和室内定位模块，下面分别论证这两个模块的选择。

1. 高度测量模块的论证与选择

方案一：采用 BMP085 气压传感器测量大气压并转换为海拔高度，把当前的海拔测量值减去起飞时的海拔值，即得飞机的离地高度。

方案二：采用 KS109 超声波传感器测量飞行器当前的飞行高度。模块发出 8 个 40kHz 的超声波脉冲，然后检测回波信号。当检测到回波信号后，模块还要进行温度值的测量，然后根据当前温度对测距结果进行校正，将校正后的结果输出。

考虑到气压传感器数据需与六轴传感器数据融合后输出，算法耗时且数据精度不高，而超声波模块虽然只适合于低空飞行，但在低空时的数据准确性很高且使用方法简单，故选择方案二。

2. 图像识别模块选择

方案：STM32F407 处理器是基于 ARM® Cortex™-M4 为内核的 STM32F4 系列高性能微控制器，其采用了 90nm 的 NVM 工艺和 ART。ART 技术使得程序零等待执行，提升了程序执行的效率，将 Cortext-M4 的性能发挥到了极致，使得 STM32 F4 系列可达到 210DMPIPS@168MHz。

OV2640 摄像头模块，采用 1/4 英寸(1in＝2.54cm)的 OV2640 百万高清 CMOS 传感器制作。具有高灵敏度、高灵活性、支持 JPEG 输出等特点，并且可支持曝光、白平衡、色度、饱和度、对比度等众多参数设置，支持 JPEG/RGB565 格式输出，可以满足不同场合需求。

综合以上两点，可以较高效率地识别本次竞赛要求的图像。

3. 室内定位模块选择

方案：PX4Flow 是一款智能光学流动传感器。传感器拥有原生 752×480 像素分辨率，计算光流的过程中采用了 4 倍分级和剪裁算法，计算速度达到 250Hz(白天，室外)，具备非常高的感光度。与其他滑鼠传感器不同，它可以以 120Hz(黑暗，室内)的计算速度在室内或者室外暗光环境下工作，而不需要照明 LED。本次利用这款光流传感器使飞机能在室内稳定地飞行与悬停。

7.2.3　实现方案

1. 定高飞行设计

超声波模块接收到数据，发给系统并与设定值进行比较、判断，进行 PID 计算后，输出值用于油门控制，以达到在指定高度平稳悬停的效果。

2. 室内悬停设计

定点悬停是在姿态控制及高度控制的基础上实现的，首先姿态控制使飞机能够平稳地飞行，而高度控制是飞行器保持在目标高度上，最后通过光流传感器获取飞行器的位移信息，根据当前累计偏移量与设定点的差值进行 PID 调节，使无人机能够平稳地悬停在设定点附近，当受到外力作用偏出时，能够及时和迅速回到设定点。

3. 图像识别设计

通过摄像头采集到图像，进行二值化处理，得到黑白像素点，然后计算出连续黑色像素的周长和面积，根据周长的平方与面积的比例，区分正方形、三角形和圆形，对于大正方形与小正方形，通过测试拟合出面积与高度和图形所占像素点个数的函数关系，在程序中计算出图形的实际面积，用于区分形状相同的不同图形。

根据图形在所采集图像中的位置，和高度数据相结合，可以计算出图形与无人机的相对偏差，指导无人机在图形正上方悬停。特别需要说明的是，所测得的相对偏差还需利用无人机的倾角进行修正，以得到准确的相对偏差。

4. 起飞与降落控制设计

一键起飞时，定时给油门一个循环增大的油门量，同时在上升过程中通过高度控制，使得飞机稳定在 3m 左右的高度。而当飞机检测到下落的图形标志点时，循环降低油门量使无人机直线安全降落，降落时，在安全、稳定的情况下尽量保证快速下降，提高无人机的降落精度。

5. 飞行循迹设计

在飞机飞到指定高度后，开始扫描地面上的图像，由于摄像头的视角有限，在 3m 高度时飞机能扫描到的范围只有不到 3m×3m，不能顾及 10m×10m 的场地上随意摆放的 4 个图形，为了能够完成题目要求，采用扇形扫描方案。

无人机在大正方形上起飞之后，首先直线进行扫描，若发现目标图形，则立刻降落，若发现其他图形，则记录无人机当前的偏航角和前进距离。无人机完成一次来回的直线扫描，回到原点，逆时针方向旋转一定的角度再次进行直线扫描，依此类推，则可以完成覆盖全场的扫描。

在一个图形上起降之后，若下一个目标已经被扫描过，则立刻从原点飞往下一个目标；否则继续向后进行扫描。

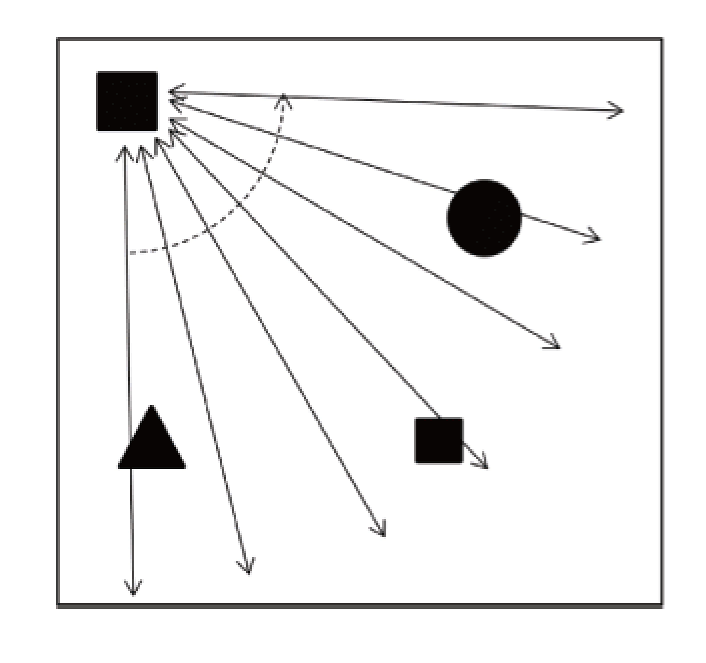


图 7.2.2 扫描示意图

无人机扫描飞行示意图如图 7.2.2 所示。

7.2.4 硬件与程序方案

1. 硬件方案

本四旋翼飞行器控制系统是由两片微处理器芯片组成，分别是由 TI 的 TM4C123G 和 STM32F407 组成。

飞行控制板主芯片为 TM4C123G，把从 STM32 处获取的数据与原有飞行控制程序融合完成题目中所有的要求。四旋翼无人机的导航芯片上装有超声波传感器、光流传感器，作用分别为定高和室内悬停。四旋翼无人机的飞行控制板上装有六轴传感器 MPU6050 模块，作用是获取角度数据，通过 PID 和姿态解算进行飞行。

图像处理模块为以 STM32F407 为核心的自主设计板，其上方还装载有 OV2640 摄像头，用于采集地面图像，采集到数据后进行一定的算法处理，将其位于图像中的坐标通过串口发给飞行控制板。

其硬件框图如图 7.2.3 所示。

2. 软件方案

软件方案同前文所述基本功能无人机的软件方案类似，仅为在其基础上，在定时器中断中加入竞赛所需程序，包括循迹算法，对于图传板所传数据的处理程序等，如图 7.2.4 所示。

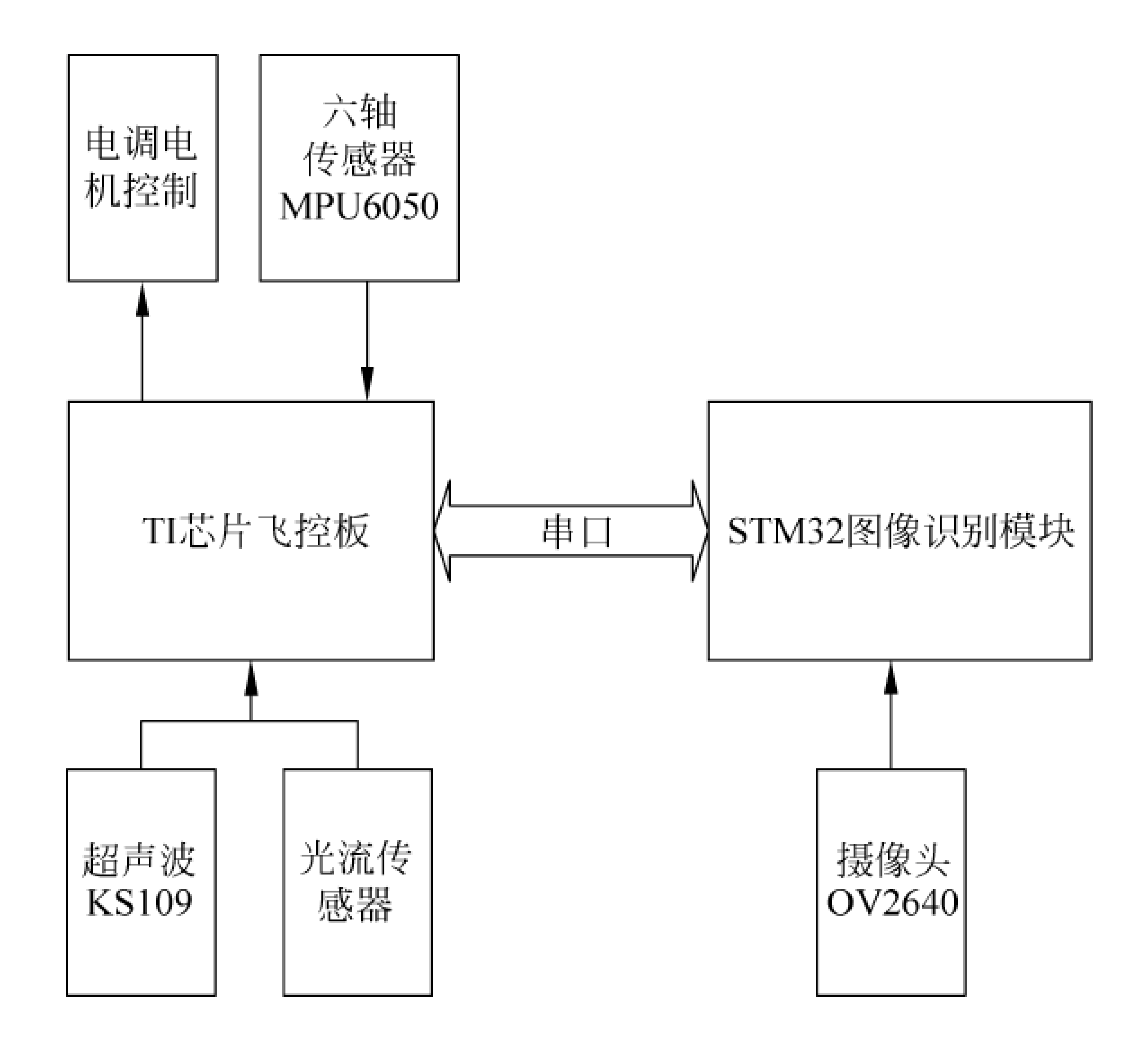


图 7.2.3　硬件框图

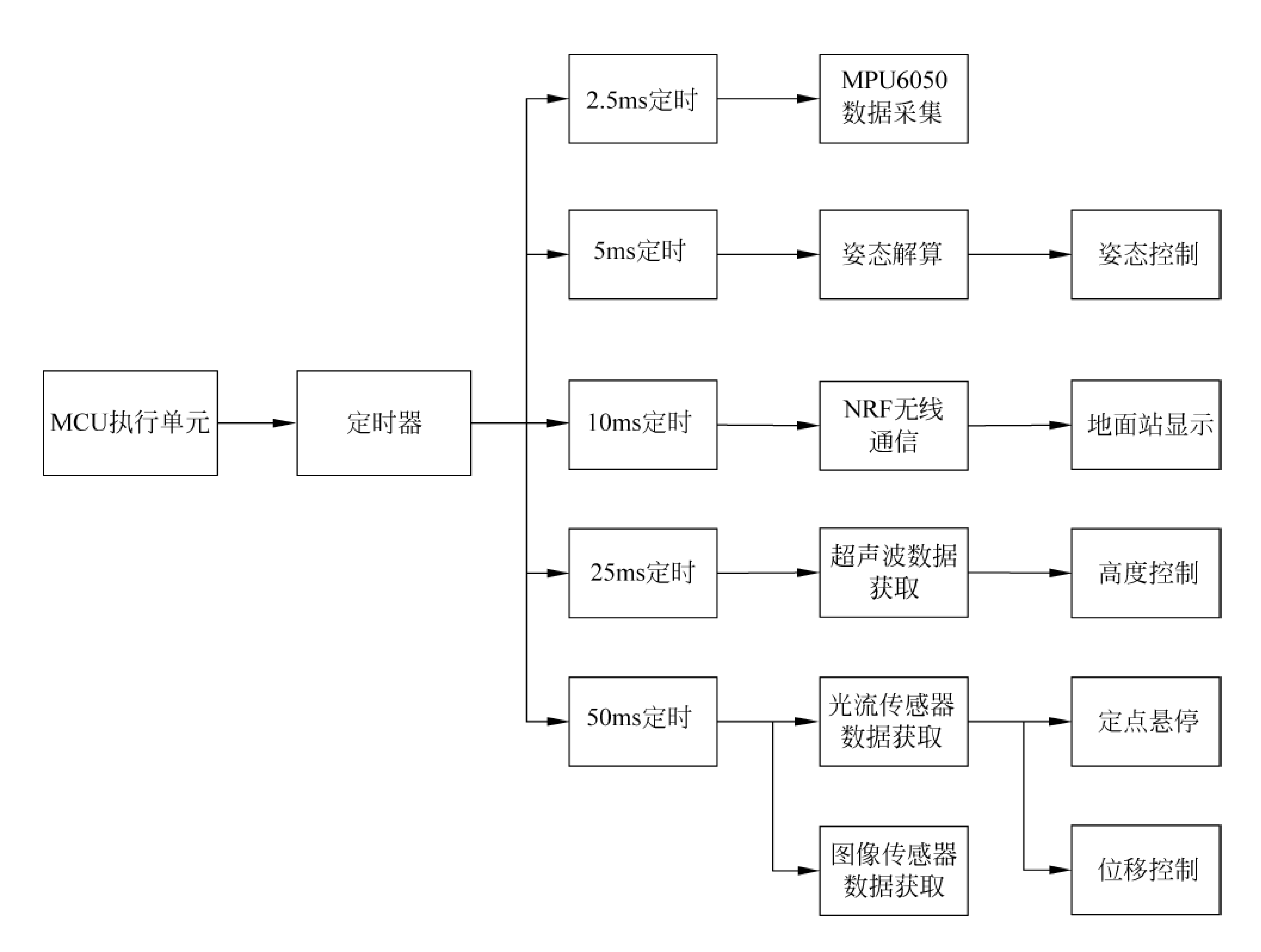


图 7.2.4　软件流程框图

7.2.5 测试

1. 测试方案

飞行器摆放在大正方形上方，无线遥控装置发出起飞命令，飞机从大正方形中心起飞，自动按照圆形、三角形、小正方形顺序飞行，在每一个标志物上完成降落和起飞，最后降落到小正方形中心。观察飞机对于每个图案识别的准确度，以及在图像上方的起飞、下降的准确性，最后还有完成整个过程的总时间。

2. 测试条件

飞行器应该在带有纹理的地面(10m×10m)上起飞，场地有随机分布的黑色图像(如大正方形、小正方形、三角形、圆形)作为指示物引导飞行器前进，室内光线以灯光为主。用秒表记录飞行的总时间，上位机记录飞机的飞行高度。

3. 测试结果

测试结果如表7.2.1所示。

表7.2.1 测试记录表

飞行测试次序	1	2	3	4	5
飞行高度/cm	约290	约290	约300	约290	约300
大正方形	√	√	√	√	√
圆形	√	√	√	√	√
三角形	√	√	√	√	√
小正方形	√	√	√	√	√
飞行时间/s	178	187	153	214	199

7.3 毕业设计作品1——目标跟踪四旋翼飞行器

7.3.1 课题概述

本课题的目的在于实现四旋翼飞行器飞行路径记忆及目标跟踪，以及即时定位与建图，实现飞行器的完全自主飞行。

7.3.2　硬件系统设计

1. 系统总体结构

系统硬件框图如图 7.3.1 所示。

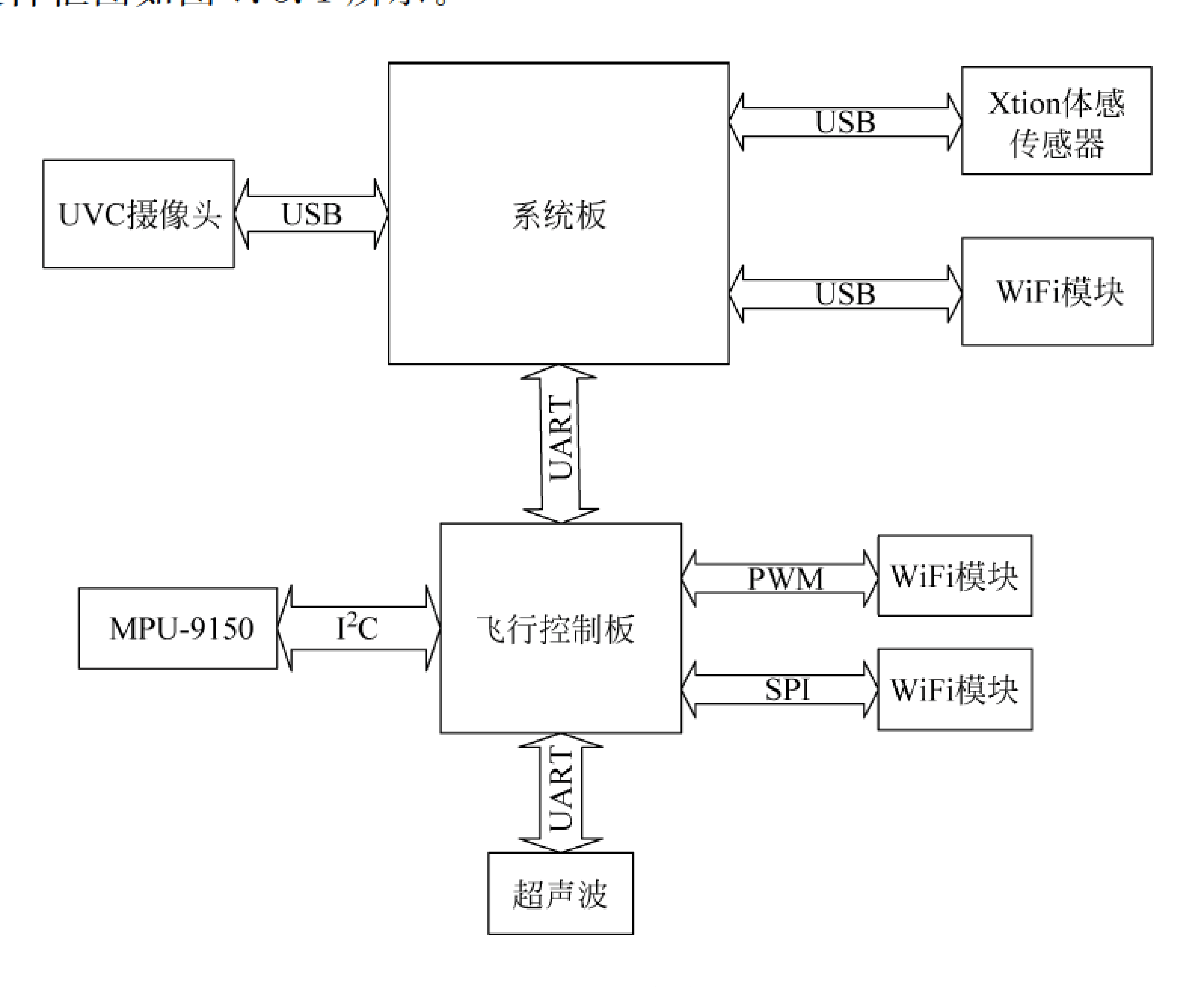


图 7.3.1　硬件框图

2. 机械结构

机械部分包括超轻机架的设计、动力系统的配置。

机械部分是整个系统基础，对于本课题的试验性质，机械部分必须是廉价的、兼顾强度、重量与可维护性的。同时机械设计时还要考虑一定的冗余性，保证拓展的可能。

3. 传感器

传感器包括获取飞行器位姿的陀螺仪、加速度传感器、超声波传感器、获取视觉信息的摄像头以及获取立体视觉的 RGBD 摄像头 Xtion。

4. 飞行控制板

飞行控制板需要连接多种传感器，如 IMU（Inertial Measurement Unit，惯性测量单元）、超声波等，来获取姿态信息，输出 PWM 波控制电子调速器，同时还要与系统板建立可靠的数据通信链路。

飞行控制板上运行一些实时性要求比较高的算法，比如飞行器的位姿解算、自动稳定控制算法等。

5. 系统控制板

系统控制板需要连接摄像头、WiFi 模块以及其他大数据量的传感器以及通信模块。系统控制板需要运行一些大计算量的算法，比如即时定位于建图(Slam)算法或者是其他图像算法。

7.3.3 飞行控制板软件系统设计

系统总体软件设计如图 7.3.2 所示。

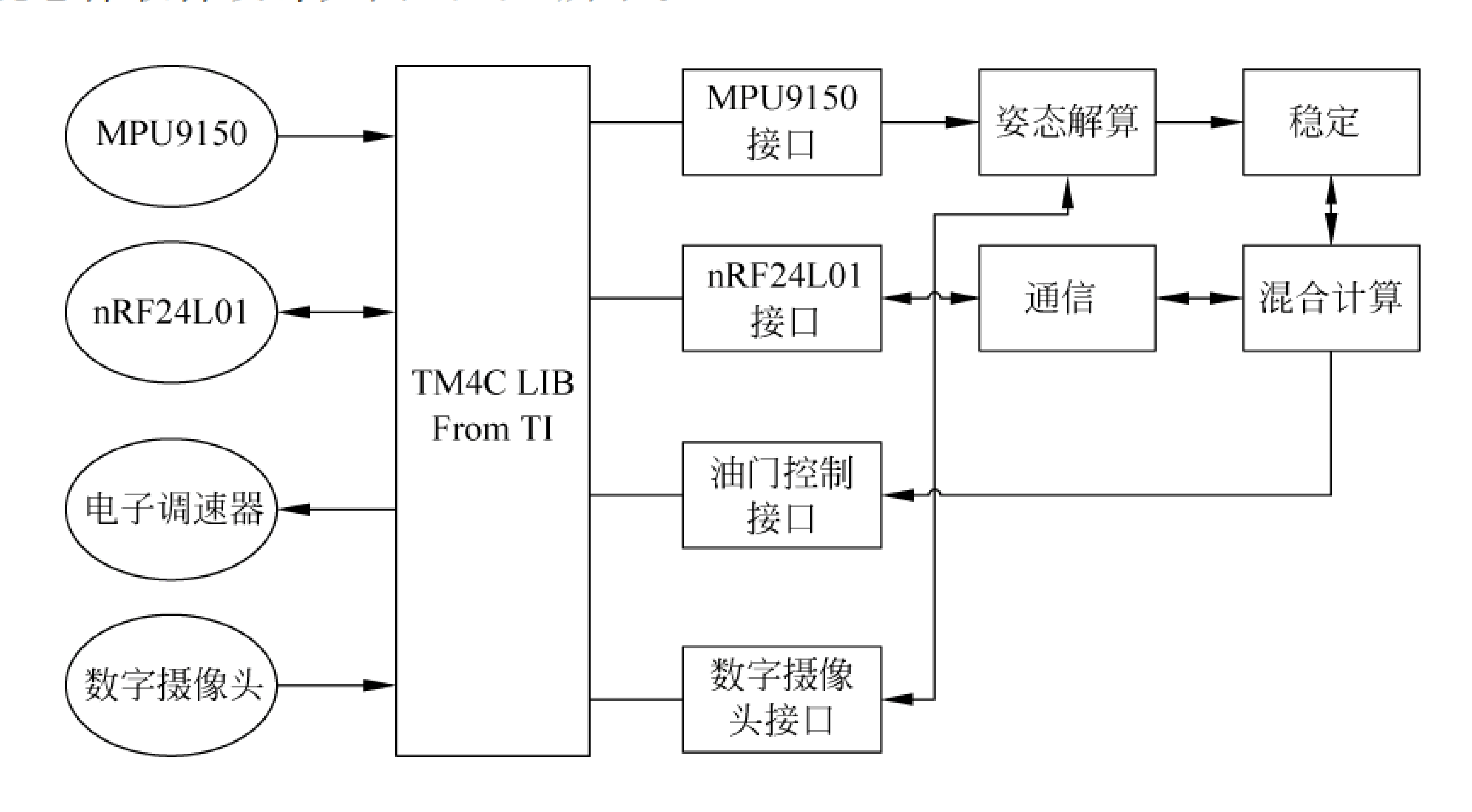


图 7.3.2 飞行控制板软件系统框图

飞行控制板软件主要分为 3 个层次。

1. 底层接口驱动

利用 TI 官方提供的底层库实现对基本芯片资源的调用，包括基本核心配置、I^2C 驱动、SPI 驱动、PWM 驱动以及 EEPROM 驱动等。

利用底层驱动实现模块化的具体硬件设备的功能，包括 MP9150 的配置以及数据读取、电子调速器的配置以及控制等。

2. 功能函数库

利用标准数学函数库结合芯片的浮点单元以及 Cortex-M4 的并行运算指令优化的 ARM 数学函数库以及姿态解算、坐标转换需要用到的线性代数库。

3. 功能应用

调用功能函数库实现具体的算法，同时通过调用硬件接口程序实现对硬件设备的安全、便捷控制。

7.3.4　系统板软件系统设计

1. ROS 操作系统

ROS(Robot Operating System)是一个机器人软件平台，它能为异质计算机集群提供类似操作系统的功能。ROS 的前身是斯坦福人工智能实验室，为了支持斯坦福智能机器人 STAIR 而建立的交换庭(Switchyard)项目。其分布式、节点化的程序结构为系统及程序的开发维护提供了极大的便利。

ROS 最大的特点在于它是一个分布式的操作系统，程序模块的最小单元为节点，节点运行在一个类似虚拟局域网的网络结构中，节点之间通过消息、服务以及行动进行数据传输。而对于单个节点在硬、软件上可以完全是异质的，ROS 的节点可以使用 C++、Python、Java Script、Octave 和 LISP 等实现，可以在大多数基于 Linux 的系统上运行，也可以在 Windows 下运行(如大多数桌面系统，以及其他嵌入式系统安卓、OpenEmbedded/Yocto 等)。

2. SVO

在移动机器人领域，即时建图与定位(SLAM)以及路径记忆与追逐被认为是实现其自主导航的基础，而自主导航又是实现机器人自主运行的基础。

长久以来，SLAM 算法一直是机器人研究的重点之一，不乏众多成熟的、效果出色的解决方案，但是当移动机器人从地面机器人转向空间飞行器时，机器人的自由度一下子从 3 个增加为了 4 个，而飞行又对处理器的功耗、体积有着较高的限制，计算性能有限，由此 SLAM 的难度一下子剧增。

因此要实现空间飞行器的 SLAM 算法需解决两个问题：简单实现三维立体视觉的传感器或者方法；高效的运动估计以及建图方法。

视觉里程计(VO)即是为解决这一问题出现的产物，通常的视觉里程计都是简单使用图像特征点来匹配，而文献[7]提到的半直接视觉里程计(SVO)方法则是一种结合了之前广泛用于激光扫描仪的直接匹配法的方案。

本系统使用 SVO 方案作为视觉定位于运动预测方案。

3. 特征集树与特征集加入

特征集树是一种用来表示特征集对应的空间数据集间的空间关系的一个数据结构，用以表示哪几个数据点集在空间上是相邻的。特征集树为一棵多叉树，其建立的过程如图 7.3.3 至图 7.3.5 所示。

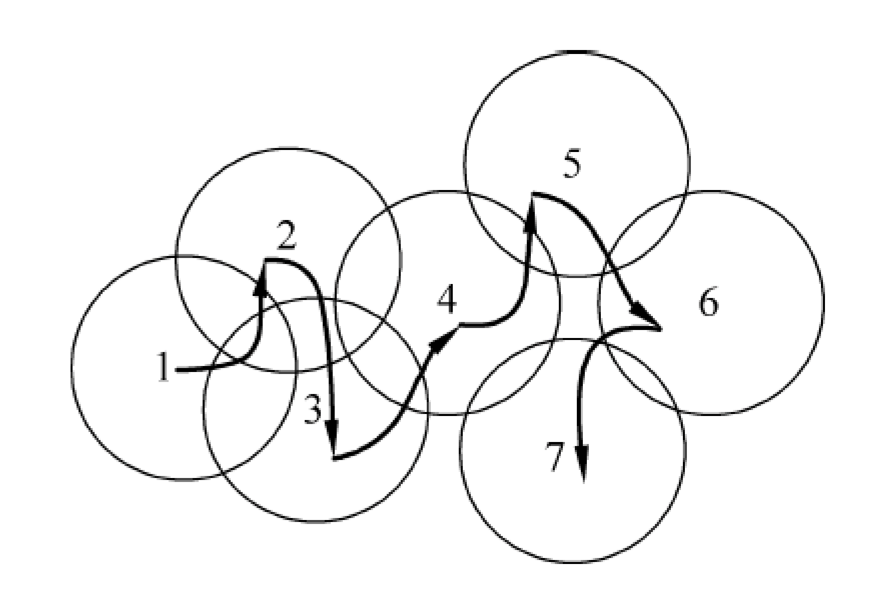


图 7.3.3 平台空间移动示意图

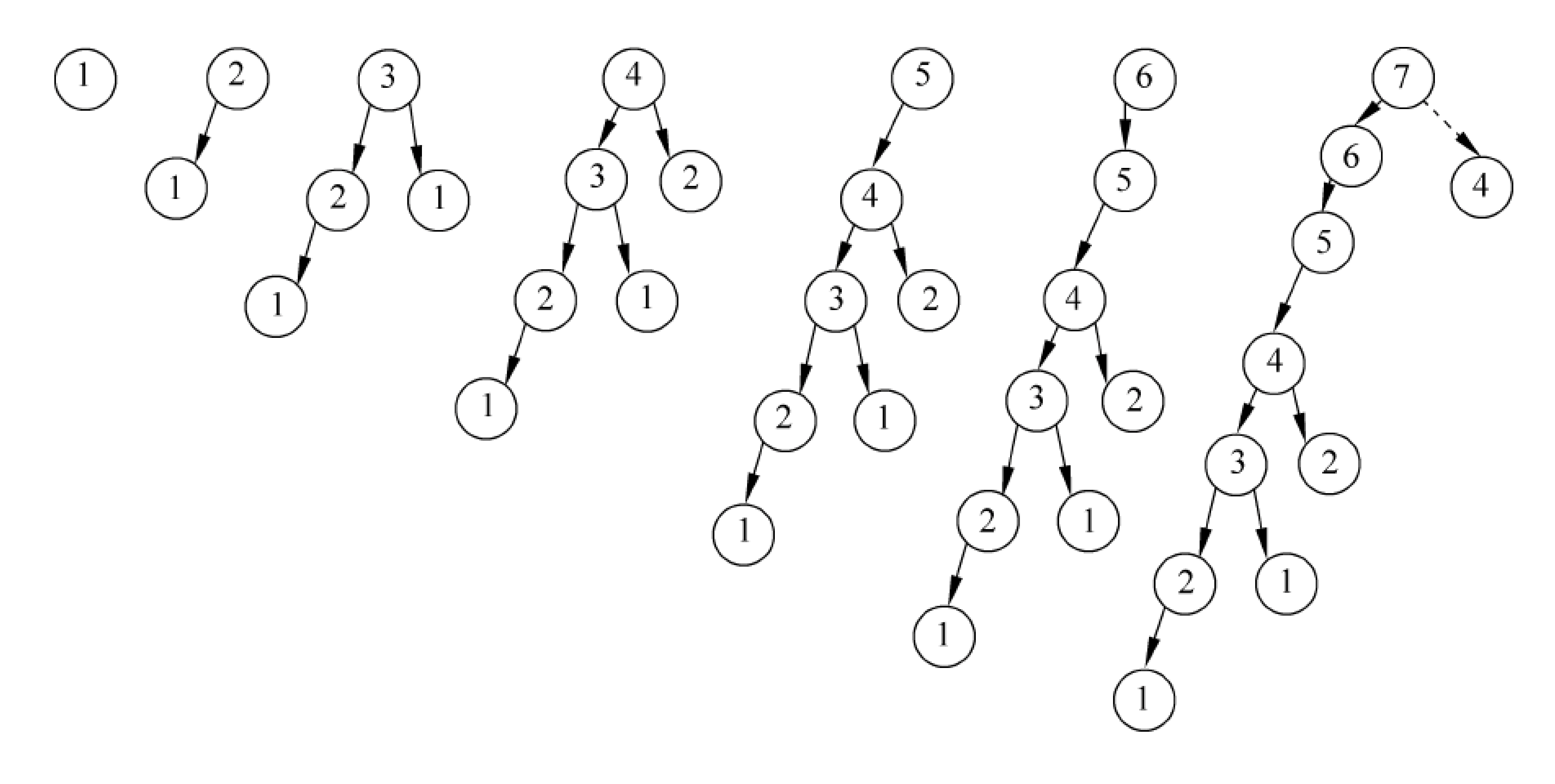


图 7.3.4 特征集树建立过程

曲线表示移动平台走过的路径，每移动一小段距离后对环境做一次观察，两次观察的距离必须保证前后两次的观察数据存在一部分重合。图中的圆圈则表示各次观察的范围。

图 7.3.4 中每个节点的第一级子节点是与其直接相关的(即存在重叠部分的特征集)，第二集及其下的子节点虽与当前节点无关，但其空间位置是与当前节点位置最近，因此稍后加入的特征集与那些节点代表的特征集相关的概率非常大。且可以认为越接近根节点概率越大。

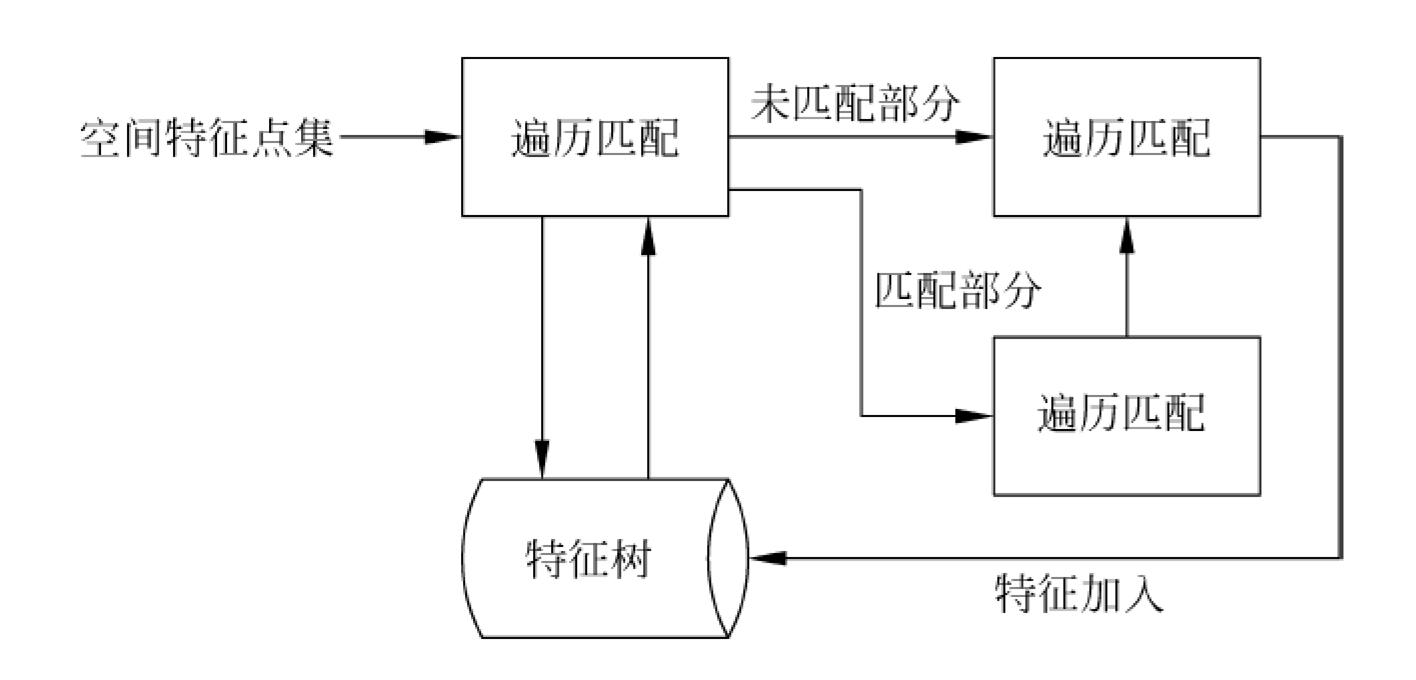


图 7.3.5　特征加入过程框图

每当有新的特征集加入，算法将从特征树的根节点出发，沿树按深度顺序进行遍历匹配，通过匹配度判断是否是直接相关，并将与其直接相关的特征点集作为加入节点的子节点。在匹配的过程中将需要加入的特征集分为已匹配与未匹配两部分，将匹配部分的特征点获取的坐标系旋转矩阵对为匹配的特征数据进行标记，再将未匹配部分的特征集作为加入的特征存在树的根节点。当然为了控制算法复杂度，必须限制遍历的深度，遍历深度越深越能够详细反映特征点集间的相互关系，当然也会导致算法复杂度的提升。

7.3.5　总结与展望

本课题涵盖了机械、控制、机器视觉、机器人学、分布式计算、嵌入式 Linux 系统、嵌入式 MCU、传感器、数据融合、数据滤波等各方面的知识与技术。覆盖面非常广，难度非常高，课题可以说是对实现四旋翼飞行器的完全自主飞行进行了一次尝试。

课题实施的过程获得了一些不错的成果，基本实现了四旋翼飞行器的路径跟随功能，基本程度上实现了四旋翼飞行器的 SLAM 算法，当然距离实现完善的自主飞行任务还有相当长的距离。

虽然目前来看地面自主移动机器人还不完美，现阶段研究飞行器的自主导航似乎有一些“还没学会走路就想着飞的”意味，但作者认为脚踏实地并不意味着不能仰望星空，四旋翼飞行器的发展虽然存在重重阻力，不少人还对其可能的应用产生了担忧，但其前景以及应用价值是无限的。

7.4 毕业设计作品 2——基于 Kinect 的穿窗和手势控制四旋翼飞行器

7.4.1 课题概述

Kinect 是一种 3D 体感摄像机，与普通摄像机相比，可以感知距离信息，在机器视觉邻域中有许多有价值的应用。本课题重点在于利用 Kinect 作为传感设备实现四旋翼飞行器的自控飞行，通过 Kinect 获取飞行器的移动状态以及深度数据，PC 作为基站部分完成图像处理以及路径计算，对四旋翼飞行器发送控制命令，实现对飞行器的跟踪定位，并且控制飞行器完成钻窗以及手势识别控制的动作。

7.4.2 硬件系统概述

整个系统硬件由两大部分组成，分别是基站部分以及飞行器部分，如图 7.4.1 所示。以下介绍主要部分。

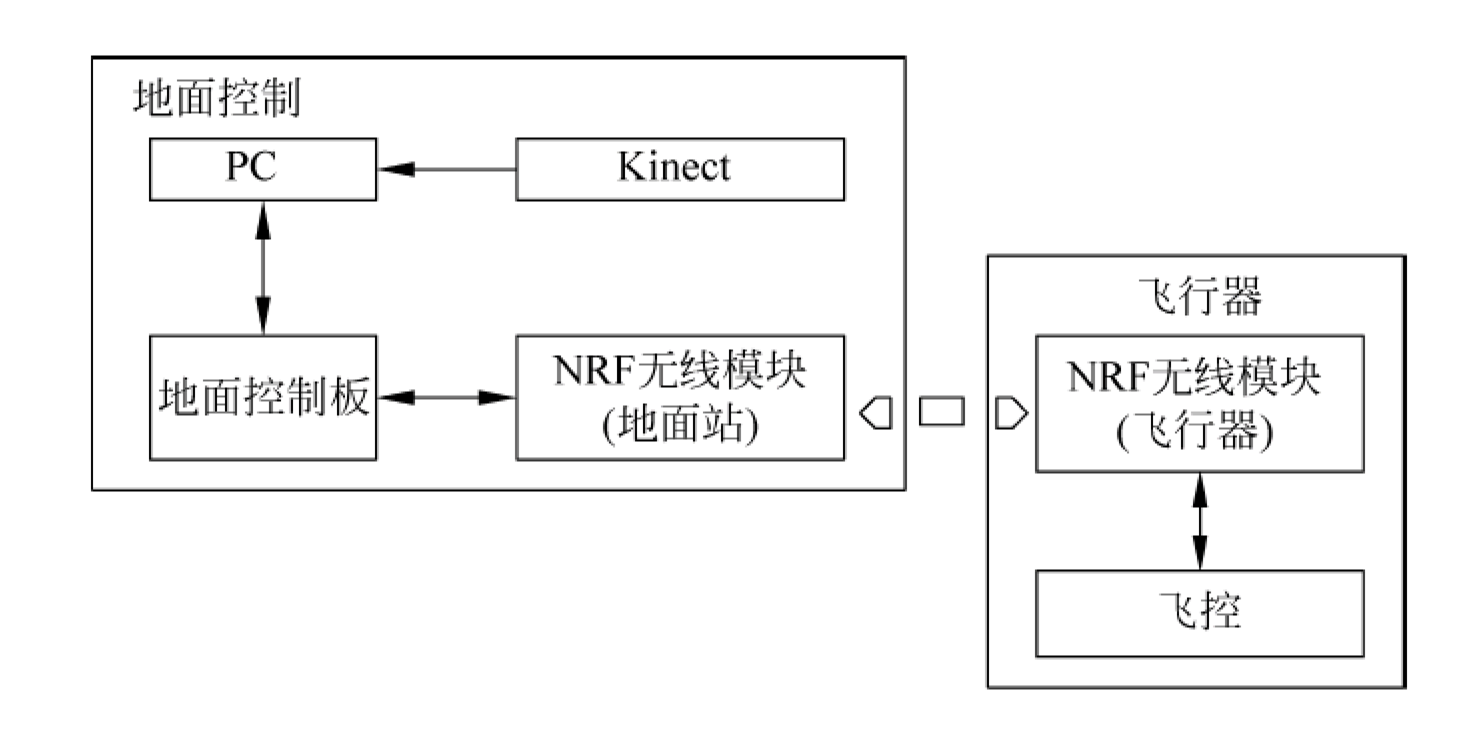


图 7.4.1 硬件结构图

1. 地面控制板

① 实现与 PC 上位机之间的交互通信。通过上位机可以设置四轴的各项参数，观测飞行器的实时状态。

② 实现飞行控制板之间的交互通信，包括传送遥控数据和控制命令数据，接收飞行器的状态数据。

③ 完成实时遥控器数据的采集。采集遥控器的 PPM 信号并解析成真正的遥控数据,实现遥控器对四旋翼飞行器的控制。

2. PC 上位机

PC 上位机的功能描述如图 7.4.2 所示。

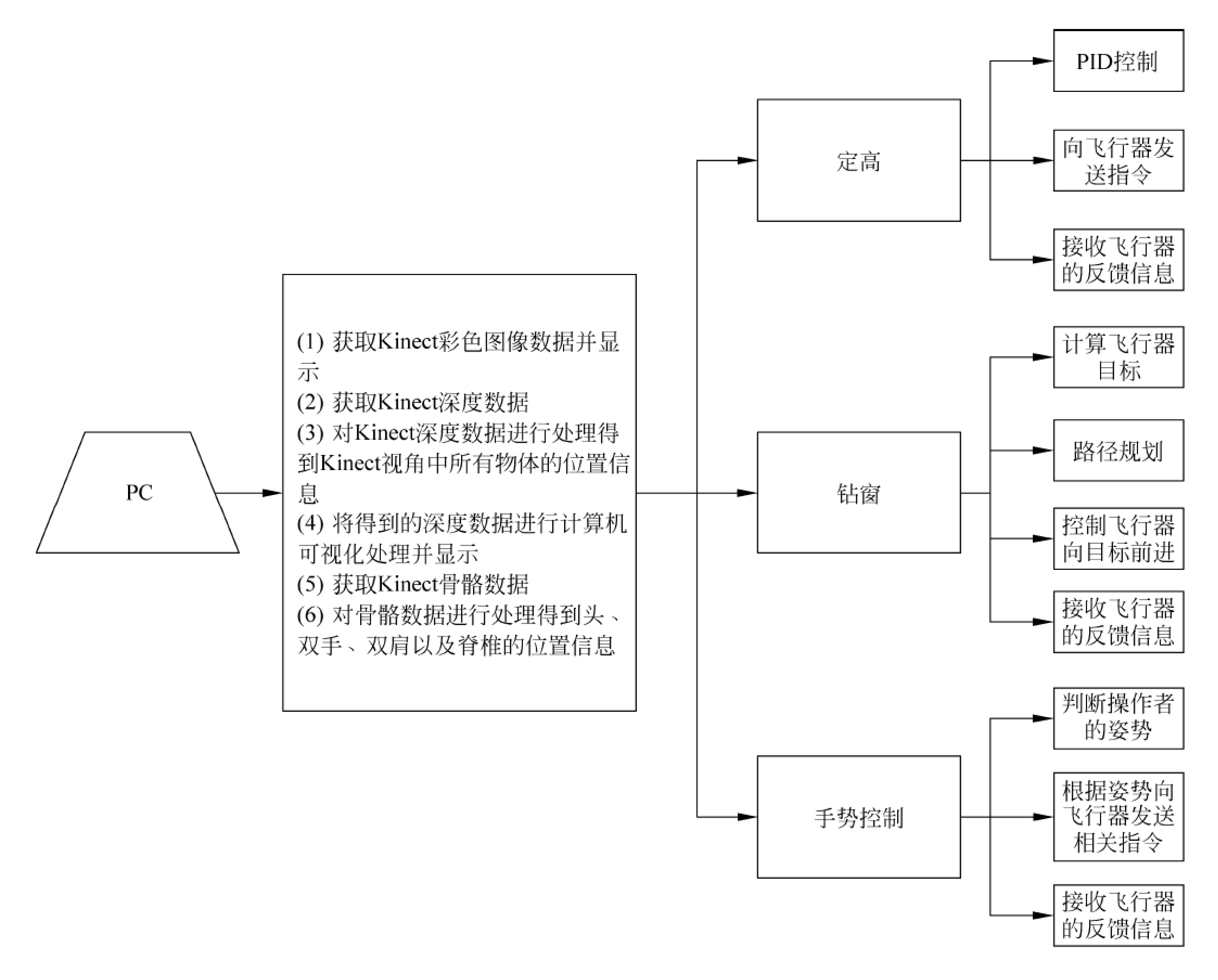


图 7.4.2　上位机功能描述

3. 飞行控制板

飞行控制板的主要功能包括以下几个。

① 飞行器传感器数据采集及处理,包括六轴传感器的姿态数据、超声波模块测量到的高度数据、光流传感器的位移数据。

② 通过电机驱动程序修改飞行控制板输出四路 PWM 波的占空比,调节 4 个电机的转速来实现对四旋翼飞行器运动状态的控制。

③ 与地面控制板的交互通信，接收地面控制站发送来的遥控器数据以及其他控制命令，同时向地面站反馈自身各项状态数据。

4. Kinect 深度摄像头

深度数据是 Kinect 的精髓，Kinect 通过发射近红外线光源来获得深度图，只要有大字形的物体，Kinect 都会去追踪，即便不开灯，Kinect 也会去追踪，Kinect 有发射、捕捉、计算视觉重现的类似过程。它的“深度眼睛”是有红外投影机和红外摄像头组合而成的，投影和接收互为重叠。

本课题中，利用深度数据，探测无人机的位置，帮助无人机稳定飞行以及完成特定动作。

在 Kinect 里面，通过 20 个关节点来表示一个骨架，具体如图 7.4.3 所示。当走进 Kinect 的视野范围时，Kinect 就可以把你的 20 个关节点的位置找到（当然你得站着），位置通过（x，y，z）坐标来表示。

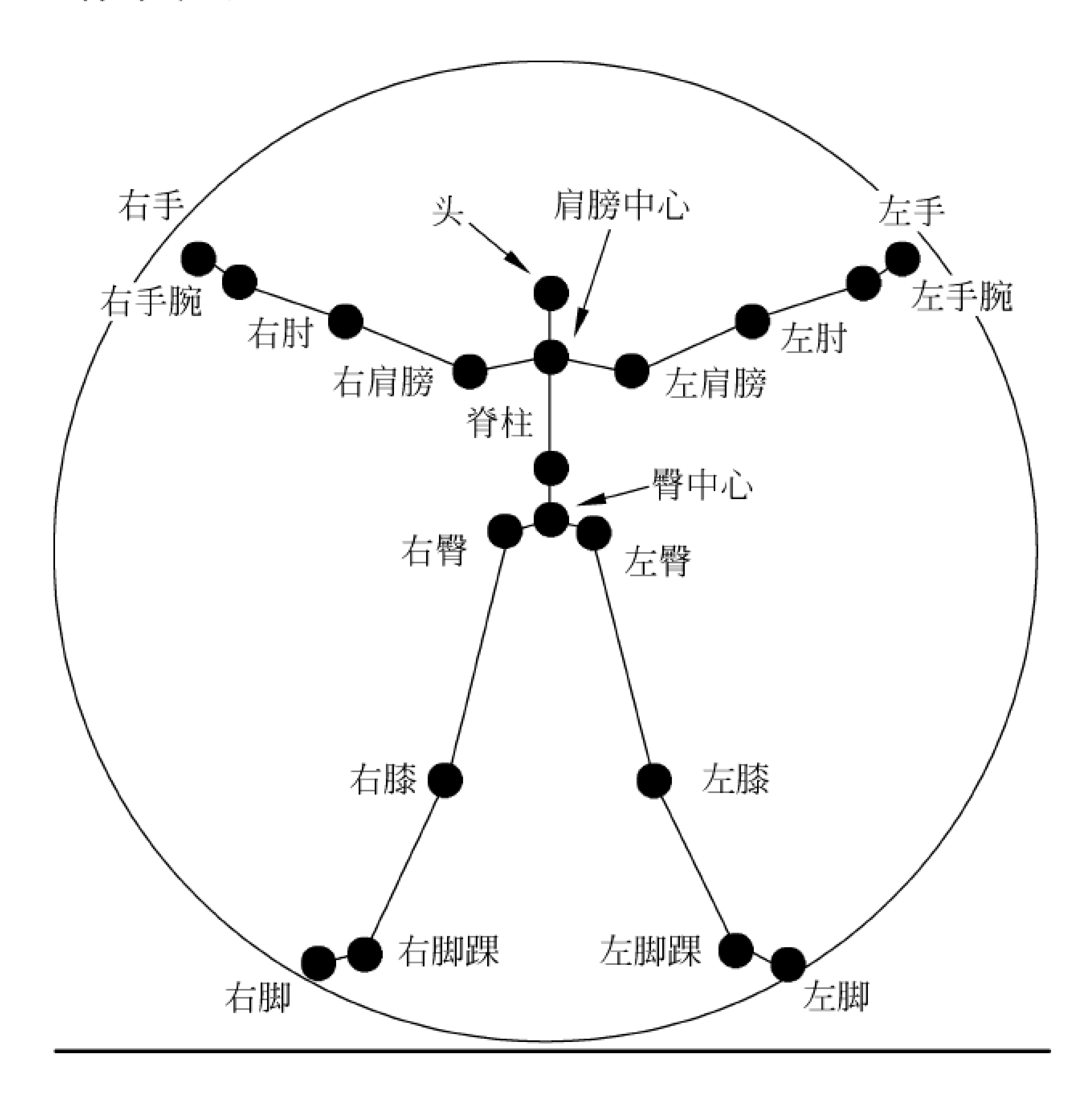


图 7.4.3 Kinect 视野中的骨骼图

通过骨架数据，利用 Kinect 提供的函数，可以识别人的特定动作，是手势控制无人机的基础，在后文中将详细介绍。

7.4.3　目标物体定位和追踪

1. 物体的识别与追踪

想要实现对四旋翼飞行器的控制，完成一系列的动作，首先要“看”到四旋翼飞行器以及 Kinect 视角中其他物体的位置，即定位，以保证飞行器在飞行过程中不会撞到障碍物，并且可以成功地控制四旋翼飞行器完成特定的任务。

1）四旋翼飞行器的识别及追踪

在对四旋翼飞行器进行定位时，把在高度阈值内的物体判断为四旋翼飞行器，然后对其进行控制。计算得到飞行器位置信息的程序如下：

```
for (int i16 = 0, i32 = 0; i16 < depthFrame.Length && i32 < depthFrame32.Length; i16++, i32
 += 4)
    {
    if ((realDepth > 2100) && (realDepth < 2450))//飞机
        {
            depthFrame32[i32 + GreenIndex] = 255;
            flyXSum += i16 % 640;
            flyYSum += i16 / 640;
            flyDSum += realDepth;
            countFly++;
        }
    }
```

程序中 flyXSum＋＝ i16％640；flyYSum ＋＝ i16/640 语句是为了把线性数据转换成 X、Y 坐标系下的坐标，以便于之后对四旋翼飞行器的位置进行更加直观地标定及控制。

由于从 Kinect 得到的数据是以像素形式出现的，所以在判断时是对逐个点进行判断的，但是由于最终需要得到的数据是四旋翼飞行器的具体位置，而根据高度阈值所判断得到的点是四轴飞行器在图像中所有的像素点，所以要对这些像素点的高度、当前高度下的 X 坐标和 Y 坐标做平均，从而确定四旋翼飞行器的位置，由于四旋翼飞行器所对应的图像中的像素点的高度大致是在同一平面上的，所以这里不考虑由于摄像头成像原理所造成的误差。

countFly 变量表示的是被判定为飞机的像素点的个数，作为之后计算当前高度下的 X 坐标和 Y 坐标的均值时的除数。

```
if (countFly > 50)
{
    flyXAverage = (int)flyXSum / countFly;
    flyYAverage = (int)flyYSum / countFly;
    flyDAverage = (int)flyDSum / countFly;
}
```

在计算完均值之后就可以得到四旋翼飞行器的当前位置信息。

2）Kinect 视角中其他物体的识别

在控制飞机自主钻窗应用中，可能在 Kinect 视角中出现的东西有四旋翼飞行器、窗以及人，而此时只要一开始通过 Kinect 标定窗高，就可以很容易地从得到的深度数据中分辨出窗、四旋翼飞行器和人。

由窗高做分割线，将和窗高高度相差不大的像素点识别为窗，然后就能根据均值算法计算出窗在 Kinect 深度图像中的位置。同时得知窗的高度后，控制飞机高度在地面到窗之间（地面离 Kinect 平面的距离由入场时进行一次标定），然后就可以定位出飞机位置。最后还需要判断的就是在 Kinect 视角中是否有人存在，以便在窗的位置改变后可以控制飞机进行下一次的运动。由于开始已经标定过窗高，认为人的最高点是一定会高于窗户的，所以就将 Kinect 视角中高于窗户的像素点判定为人。

```
if (realDepth < 2000)//窗户
{
    depthFrame32[i32 + RedIndex] = 255;
    windowXSum += i16 % 640;
    windowYSum += i16 / 640;
    countWindow++;
}
if ((realDepth > 2100) && (realDepth < 2450))//四旋翼飞行器
{
    depthFrame32[i32 + GreenIndex] = 255;
    flyXSum += i16 % 640;
    flyYSum += i16 / 640;
    countFly++;
}
if (realDepth < 1500) //人
{
    depthFrame32[i32 + GreenIndex] = 255;
    depthFrame32[i32 + RedIndex] = 255;
    countPeople++;
}
```

同样地，在得到这些像素点后，会通过计算均值的方法求出可以使用的物体的具体位置，在得到这些有效位置之后再进行路径规划以及控制四旋翼飞行器飞行等动作。

2．四旋翼飞行器的定高及定位

在识别出四旋翼飞行器的位置之后，便可以利用上位机和四旋翼飞行器通信，个人对其进行控制，保证四旋翼飞行器的平稳飞行以及实现四旋翼飞行器的定高及定位。

由于 Kinect 在采集数据时精度不够以及噪声干扰，所以对四轴飞行器识别时会存在强烈的抖动，于是就要对得到的数据进行消抖处理。

首先，把当前帧得到的四旋翼飞行器的位置信息和上一帧进行比较，x、y、d 分开比较，如果 x、y 的差距不超过 15 像素，d 的差距不超过 20mm，就判定四旋翼飞行器在两帧间隔的这段时间内没有发生位置的改变，这里以深度这一维度为例：

```
if ((flyDAverage > flyDaverage_lastframe - 20) && (flyDAverage < flyDaverage_lastframe +
20))
{
    flyDAverage = flyDaverage_lastframe;
}
else if (flyDAverage <= flyDaverage_lastframe - 20)
{
    flyDAverage = flyDAverage + 19;
}
else if (flyDAverage >= flyDAverage + 20)
{
    flyDAverage = flyDAverage - 19;
}
```

如果当前帧得到的四旋翼飞行器位置和上一帧比较之后发现相差超过所设定的值，那么就认为四旋翼飞行器的位置发生了变化。同样，以深度为例，无论是高了 20mm 或是低了 20mm，飞机当前位置信息都会被重新计算，但是不会直接就将当前所得到的飞机位置信息作为最终结果，因为这样做会产生跳变。程序的做法是将上一帧得到的飞机位置的点看作长方体盒子的中心，在当前帧得到的数据超出盒子的范围之后，就会推动盒子向当前帧所得数据的方向移动，当新盒子的边缘碰到当前帧所得四旋翼飞行器的位置时便停止移动，将新盒子的中心坐标判定为四旋翼飞行器当前所在的位置坐标。

由于得到的四旋翼飞行器的位置坐标所处的坐标系是深度坐标系，即 d 代表深度，x

和 y 分别代表得到图像的像素值而非真实的三维坐标中的 x 和 y 坐标，所以这个坐标系并不是日常所熟悉的三维坐标系，所以需要将坐标系进行转换。在具体实现过程中，对 Kinect 所得到的对应深度的图像进行标定，然后根据原理推导出相应的转换公式，得到的坐标与实际测量值之间的误差不大，可以忽略。

在成功地对四旋翼飞行器进行识别以及追踪之后，就需要利用控制原理对四旋翼飞行器进行控制了。

设定好了目标高度和目标位置，测量目标值与实际值的误差，通过上位机和下位机之间的通信不断地控制调整 4 个螺旋桨的转速，使目标值与实际值之间的误差不断减小，使实际值达到目标值。整个系统是一个反馈控制系统，为了避免复杂的精确动力学建模，在这里选用 PID 控制器，定高流程框图如图 7.4.4 所示。

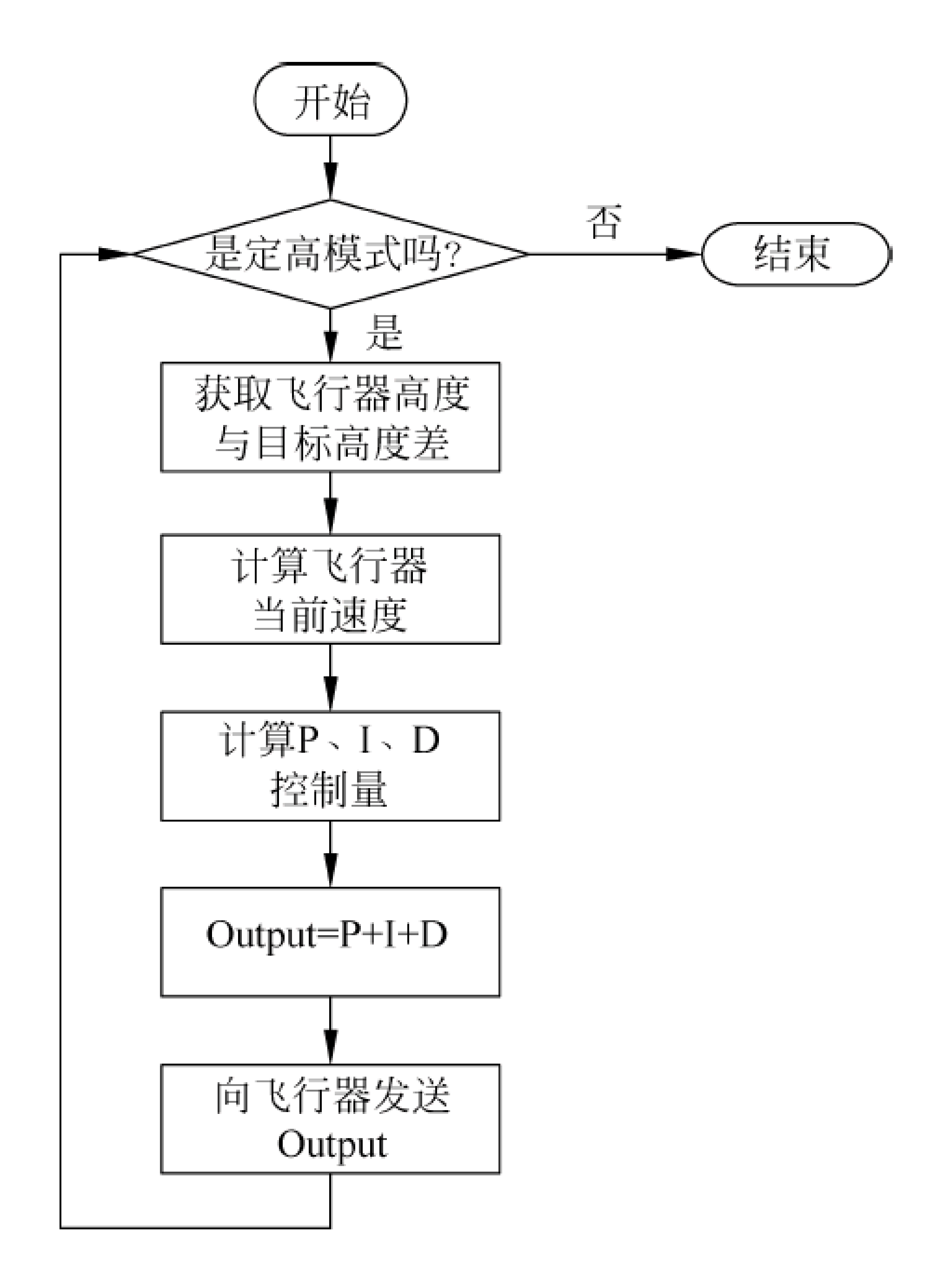


图 7.4.4　定高流程框图

由于对 3 个轴的控制相对独立且控制思路与方法也相差不大，所以以定高为例进行阐述。

高度数据已经通过 Kinect 得到，所以需要做的就是对四轴飞行器进行 PID 控制，这里直接控制的量是四旋翼飞行器的油门，所得到的数据是飞行器的位置，需要飞行器到达想要控制的目标高度，其中比例项 P 是飞行器目标位置与当前位置的差，积分项 I 是飞

行器目标位置与每一帧所得的位置差的累积，微分项 D 就是飞行器的速度，即当前帧位置与上一帧位置的差。

3. Kinect 视角中物体的位置关系转换

由于从 Kinect 上所获取到的深度数据是像素点所在的那个和 Kinect 所在平面平行的平面和 Kinect 所在平面之间的距离，以及摄像头本身的成像原理等原因会造成得到的像素点在当前平面上的 x、y 坐标并不是其在整个三维直角坐标系中的 x、y 分量[5]，于是就造成了一个问题：如果直接用得到的像素点所在平面的 x、y 分量计算两个不同高度的物体的水平距离，即利用公式 $D^2=(x_1-x_2)^2+(y_1-y_2)^2$ 所得到的距离并不是实际的两个物体之间的水平距离。

图 7.4.5 所示的斜线 kC 上的所有点映射到 Kinect 的传感器上只是一个点而已，也就是说，如果在地面上放着两个不同高度的物体，它们的顶点恰好在这条斜线上，那么按照之前的距离计算方法得到的这两个物体的水平距离就是零，显然这个结果是错误的。

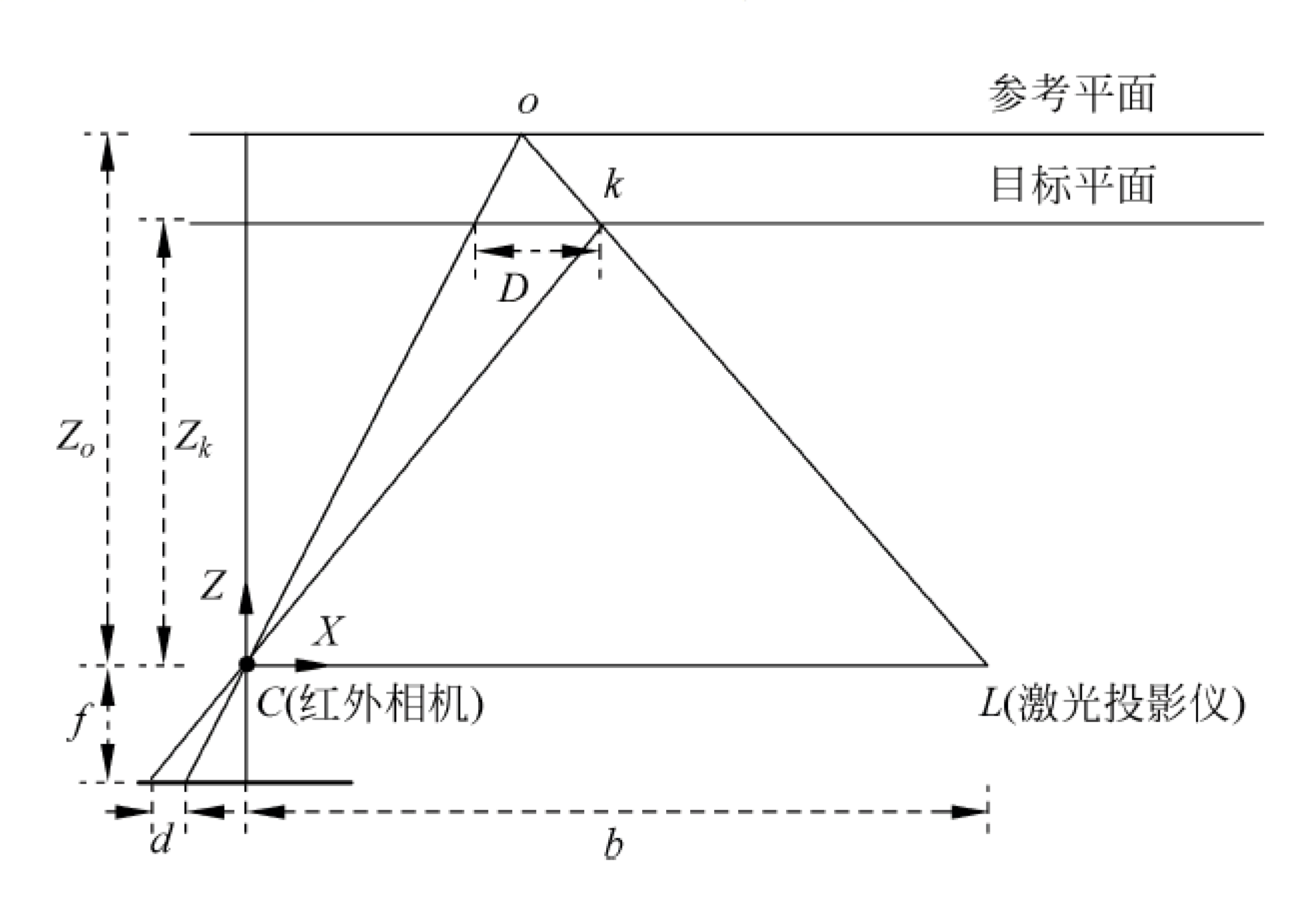


图 7.4.5　摄像头成像示意图

想要得到正确的水平距离就必须将 Kinect 得到的深度数据坐标转换成三维直角坐标系下的坐标，然后再利用公式 $D^2=(x_1-x_2)^2+(y_1-y_2)^2$ 计算出两个物体的水平距离，此时这个水平距离才是实际的水平距离。

这个坐标转换的公式可以通过相似三角形的比例规律得出，这里不作详述。

这个问题的另一个解决方案就是利用微软所提供的 Kinect for Windows SDK 中的转换类。

7.4.4 路径规划及控制

1. 自主钻窗

图 7.4.6 所示,此应用要实现的是通过 Kinect 识别出四旋翼飞行器以及窗户的位置后,通过 PC 进行路径规划,然后向四旋翼飞行器发出控制指令实现四旋翼飞行器的钻窗功能,同时在窗户位置发生改变后,通过 Kinect 的观察,可以实现对窗户的再次定位以及通过 PC 控制四旋翼飞行器再次完成钻窗的动作。

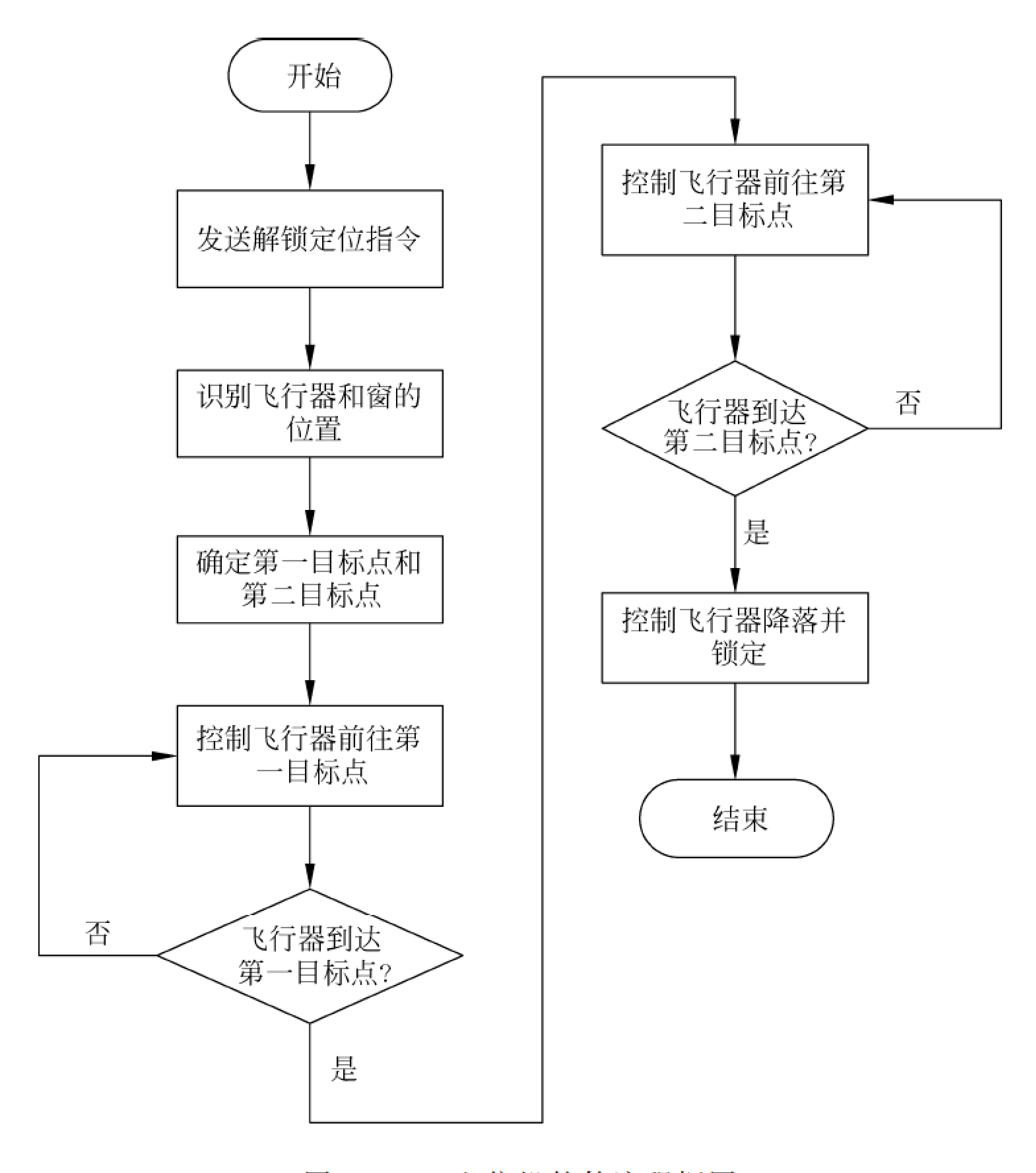


图 7.4.6 上位机软件流程框图

1) 一些状态变量的设置及作用

在对四旋翼飞行器以及窗户进行定位时,四旋翼飞行器当前的状态、人进入的情况、是否完成钻窗都需要利用一些状态变量来记录。

```
bool startFindWin = false;                  //开始寻找窗户
bool emegStop = false;                      //紧急停机
bool isFlying = false;                      //表示飞机正在飞行
bool theFirstTimeCalprePos = false;         //为了保证计算 prePos 位置只做一次
bool getToPrePos = false;                   //是否到达第一个目标位置
bool getToPrePossymmetry = false;           //是否到达目标最终位置
bool isPosCtrl = false;                     //表明已经开始定位
bool isPeopleIn = false;                    //是否有人进入 Kinect 视角中
```

设定这些标志位之后，程序就可以根据相应的标志位状态来判断是否要做接下去的操作。例如，当有人进入 Kinect 视角时就会控制四旋翼飞行器不做动作，只有等人出去之后，才继续对四旋翼飞行器发送指令，完成相应任务。

所用到的标志位在后面内容中会详细阐述其作用。

2）窗户方向的识别

7.4.3 节中已经介绍了如何对 Kinect 视角中的物体进行定位，但是由于四旋翼飞行器钻窗这一举动需要从窗的一侧穿过到达另一侧，所以还需要做的就是判断窗户的朝向（本课题中的方法仅限于判断窗户是和 Kinect 平行还是与 Kinect 垂直）。

为了解决这个问题，本程序采取的方法是在窗顶一边放置一个标志物（下文中简称“小旗”），通过窗户位置以及小旗的位置，可以得到和窗户平行的向量（这里不考虑 z 轴，仅讨论和地面平行的二维平面），得到这个向量后，窗户的朝向就能很方便地被确定：

```
deltaX = flagXAverage - windowXAverage;
deltaY = windowYAverage - flagYAverage;
```

根据 deltaX、deltaY 就可以计算出窗户的朝向，从理论上来说，如果 deltaX=0，就说明窗户是和 y 轴平行的，即窗户的朝向与 x 轴平行；同理，deltaY=0，说明窗户是和 x 轴平行的，即窗户的朝向与 y 轴平行。

在得到窗户的朝向之后，就需要进一步确定四周飞行器的目标位置，在本程序中，把四旋翼飞行器的目标分成两个，即窗户前的一点和窗户后的一点，如图 7.4.7 所示。

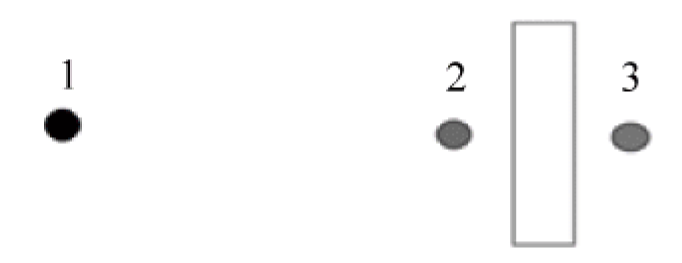


图 7.4.7 四旋翼飞行器及目标位置示意图

在图 7.4.7 中，1 号点代表四旋翼飞行器，2 号点即窗户前的一点（即在四旋翼飞行器这一侧的目标点），3 号点即窗户后的点（即在四旋翼飞行器另一侧的目标点），也是四旋翼飞行器最终的目标点。

在程序中表现为对 deltaX 和 deltaY 的判断：

```
if (deltaX < 35 && deltaX > - 35)//NS,由窗户的矢量方向算出
{
     prePosX = (flyXAverage - windowXAverage >= 0) ? (windowXAverage + 100) :
(windowXAverage - 100);
    prePosY = windowYAverage;
    prePossymmetryX = (flyXAverage - windowXAverage >= 0) ? (windowXAverage - 100) :
(windowXAverage + 100);
    prePossymmetryY = windowYAverage;
    dirWinS2N = true;
    dirWinE2W = false;
}
else if (deltaY < 35 && deltaY > - 35)//EW
{
    prePosX = windowXAverage - 20;
     prePosY = (flyYAverage - windowYAverage >= 0) ? (windowYAverage + 100) :
(windowYAverage - 100);
    prePossymmetryX = windowXAverage - 20;
    prePossymmetryY = (flyYAverage - windowYAverage >= 0) ? (windowYAverage - 100) :
(windowYAverage + 100);
    dirWinS2N = false;
    dirWinE2W = true;
}
```

在这段代码中，prePosX 和 prePosY 代表第一目标点，prePossymmetryX 和 prePossymmetryY 代表第二目标点，也是最终目标点的 x、y 值。dirWinS2N 和 dirWinE2W 分别表示对窗户朝向的判断结果。

在识别出四旋翼飞行器的位置以及窗户的位置并且判断出窗户的朝向之后，就要进行具体的对四旋翼飞行器的路径规划。

为了保证四旋翼飞行器在钻窗的过程中不碰到窗户的左、右两侧，程序首先会控制四旋翼飞行器到达和窗户同一直线上的点，然后再控制四旋翼飞行器向前飞至目标 1 点，在确定判断出四旋翼飞行器到达目标 1 点后，控制四旋翼飞行器向前穿过窗户，到达所识别出的最终目标点，其过程如图 7.4.8 所示。

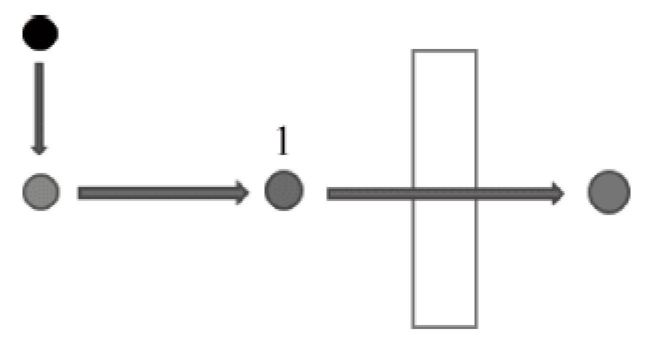


图 7.4.8 四旋翼飞行器飞行路径示意图

3）飞行方向及距离控制

在识别出四旋翼飞行器位置、窗户位置以及规划出简单的路线之后，所要做的就是对四旋翼飞行器进行控制，使其能按照所规划的路径运动，最终到达目标位置，实现钻窗动作。

由于 Kinect 摄像头以及其他原因会导致数据中存在一些噪声，从而会导致在四旋翼飞行器定位时，即使飞行器稳定地停在一个点上，也会被判定为在这个点周围晃动，所以在对四旋翼飞行器是否到达指定目标点的判断中，程序留出了误差的容限值，认为四旋翼飞行器定位在以目标点为中心的边长为 25mm 的方块内都判断为到达目标。

控制流程图如图 7.4.9 所示。

```
deltaFlyX2pre = flyXAverage - prePosX;
deltaFlyY2pre = flyYAverage - prePosY;
deltaFlyX2sym = flyXAverage - prePossymmetryX;
deltaFlyY2sym = flyYAverage - prePossymmetryY;
```

其中 deltaFlyX2pre 和 deltaFlyY2pre 是四旋翼飞行器与第一目标点的 X 轴与 Y 轴方向上的距离，deltaFlyX2sym 和 deltaFlyY2sym 是四旋翼飞行器与最终目标点的 X 轴与 Y 轴方向上的距离。

在得到这些变量后，下面一步就可以对四旋翼飞行器进行具体位移量的控制了。在进行位移控制时，要做的就是控制飞行器到达之前所判断出的目标点，首先，控制飞行器在 X 方向上到达目标点的 x 坐标，然后再控制飞行器在 Y 方向上到达目标点的 y 坐标，最后对得到的飞行器位置进行判断，判断出飞行器是否到达目标点，由于噪声以及飞行器本身的微小误差，所以目标点并不是单纯的一个点，而是那个点所辐射出的一个正方形区域，只要飞行器到达这个区域内，就认为飞行器到达了目标。

```
if (dirWinE2W == true && dirWinS2N == false && isFlying == false)
{
    if (getToPrePos == false)
    {
        if (deltaFlyX2pre >= 25)
        {
            if (deltaFlyX2pre >= 120)
            { sp.Write(TxBufferFrameMovL60, 0, 32); }
            else if (deltaFlyX2pre >= 60)
            { sp.Write(TxBufferFrameMovL32, 0, 32); }
            else
            { sp.Write(TxBufferFrameMovL15, 0, 32); }
        }
        else if (deltaFlyX2pre <= -25)
```

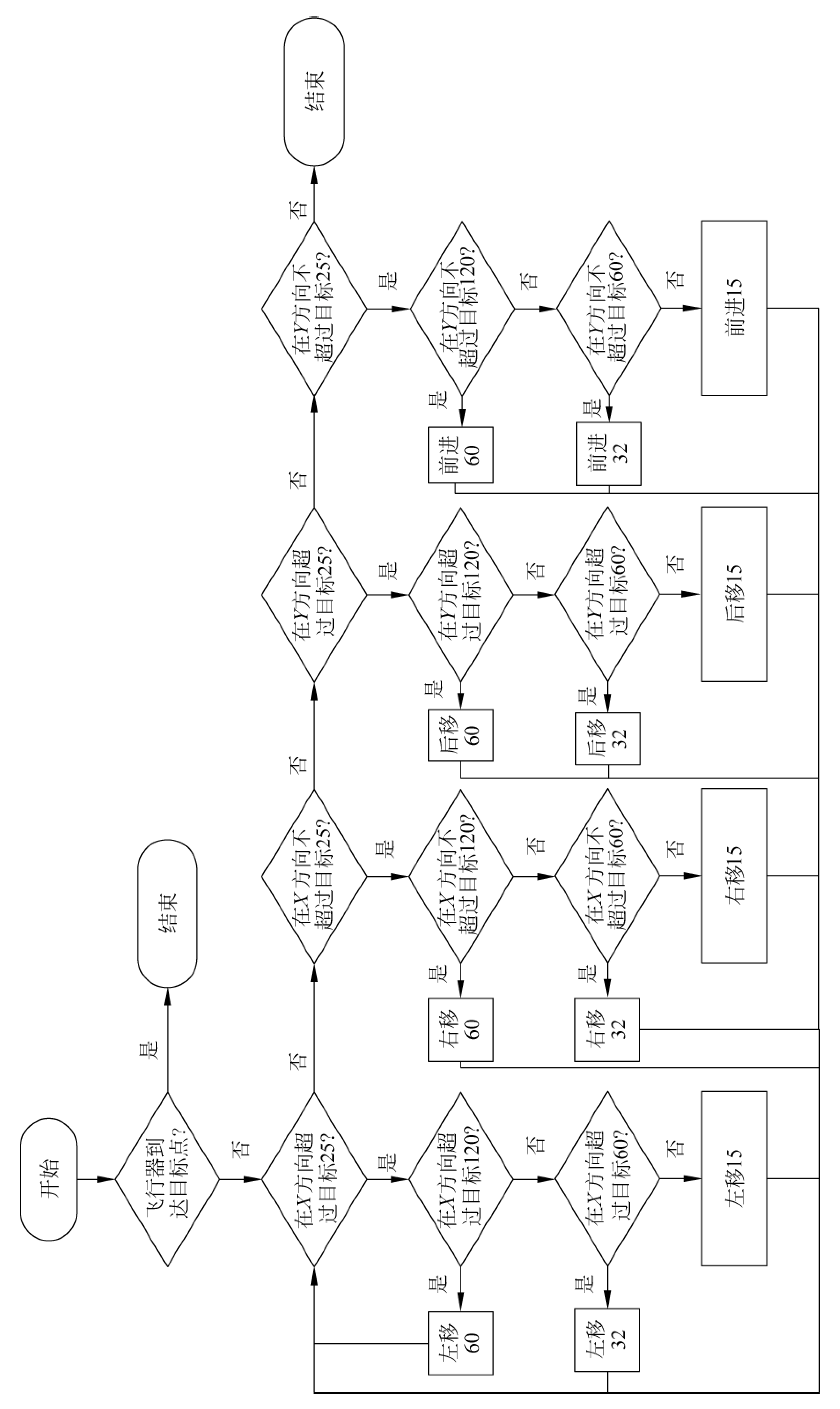


图 7.4.9 控制飞行器寻找目标流程图

```
        {
            if (deltaFlyX2pre <=  -120)
            { sp.Write(TxBufferFrameMovR60, 0, 32); }
            else if (deltaFlyX2pre <=  -60)
            { sp.Write(TxBufferFrameMovR32, 0, 32); }
            else
            { sp.Write(TxBufferFrameMovR15, 0, 32); }
        }
        else if (deltaFlyY2pre >=  25)
        {
            if (deltaFlyY2pre >=  120)
            { sp.Write(TxBufferFrameMovB60, 0, 32); }
            else if (deltaFlyY2pre >=  60)
            { sp.Write(TxBufferFrameMovB32, 0, 32); }
            else
            { sp.Write(TxBufferFrameMovB15, 0, 32); }
        }
        else if (deltaFlyY2pre <=  -25)
        {
            if (deltaFlyY2pre <=  -120)
            { sp.Write(TxBufferFrameMovF60, 0, 32); }
            else if (deltaFlyY2pre <=  -60)
            { sp.Write(TxBufferFrameMovF32, 0, 32); }
            else
            { sp.Write(TxBufferFrameMovF15, 0, 32); }
        }
        else
        {
            getToPrePos = true;
            sp.Write(clearFlowData, 0, 32);
        }
    }
    if (getToPrePos == true && getToPrePossymmetry == false)
    {
        …
    }
}
```

其中 if (dirWinE2W == true && dirWinS2N == false && isFlying == false) 这句语句是在判断窗户的朝向以及四旋翼飞行器是否在进行定位，是否可以由上位机对其进行控制，若值是 false 则说明没有对四旋翼飞行器在进行定位，可以由上位机对其进行控制。

if (getToPrePos == false)和 if (getToPrePos == true && getToPrePossymmetry ==

false)是在判断四旋翼飞行器是否到达第一个目标位置，如果没有到，那么首先要控制飞行器到达第一个目标位置，如果到了目标位置 1，却没有到达最终目标，则由上位机继续对飞行器进行控制，使其到达最终的目标位置。

在 if (getToPrePos == true && getToPrePossymmetry == false)后所省略的内容是上位机对飞行器进行位移控制的过程，和 if (getToPrePos == false)下的语句作用类似。

由于窗户的朝向有两种情况，所以需要分类讨论，以上代码只完成了窗户朝向为 dirWinE2W 的情况，在另一种情况下(dirWinS2N)，上位机对飞行器的控制方法与其类似，所以这里就不再赘述。

在控制四旋翼飞行器钻窗过程中，遇到的主要难题有以下 3 个。

① 在判断窗的状态时会遇到问题，即窗口朝向的问题，在本课题中，窗口朝向可以有两个，一个是水平的，一个是垂直的，但是仅仅靠 Kinect 所读取到的有关窗的数据很难判断出窗口的朝向。

② 由于四旋翼飞行器本身的晃动导致对飞行器是否到达目标点的判断造成了干扰。

③ 由于是根据高度来判定物体位置的，所以当有人进入场景中时，就会造成对窗以及飞行器位置判断的误差，所以对人是否在画面中的判断也是不可缺少的。

针对这 3 个问题也采取了一些相应的解决方案。

① 对于这个问题，采用了在窗户一角放上一个标志物(如图 7.4.10 所示)的方法来解决，通过判断标志物和窗的相对位置便能很好地判断出窗的朝向。

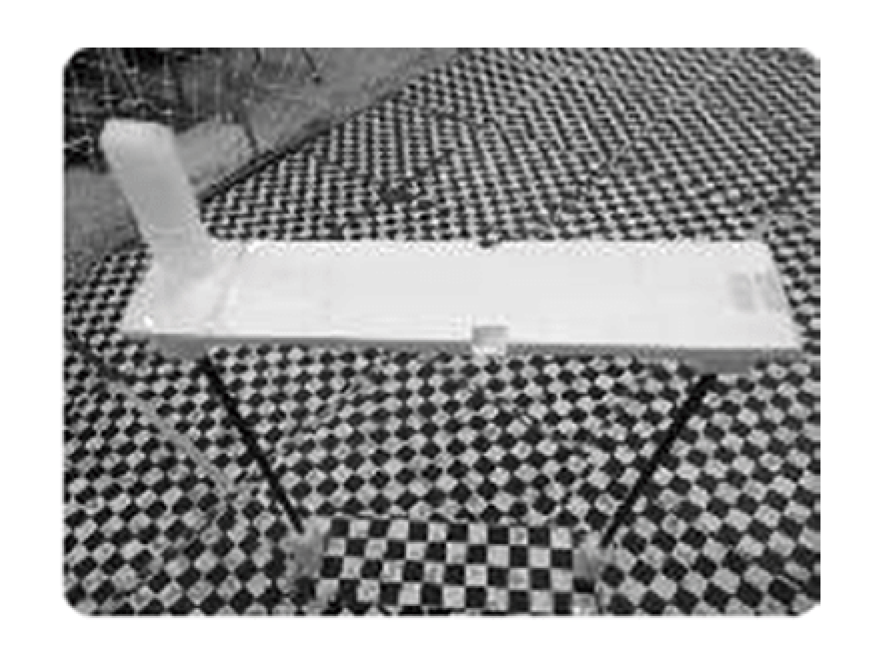


图 7.4.10　窗户以及窗户一角的标志物

② 通过“盒子”滤波可以解决飞行器晃动的问题。

③ 通过新增加标志位的方法来对是否有人在画面中进行判断并且将判断结果赋值给标志位，从而在下一帧中可以得到人进出画面的数据。

图 7.4.11 是钻窗程序所接收到 Kinect 的彩色图像数据以及深度图像数据，并对其进行计算机可视化后得到的结果，如图 7.4.11 所示。

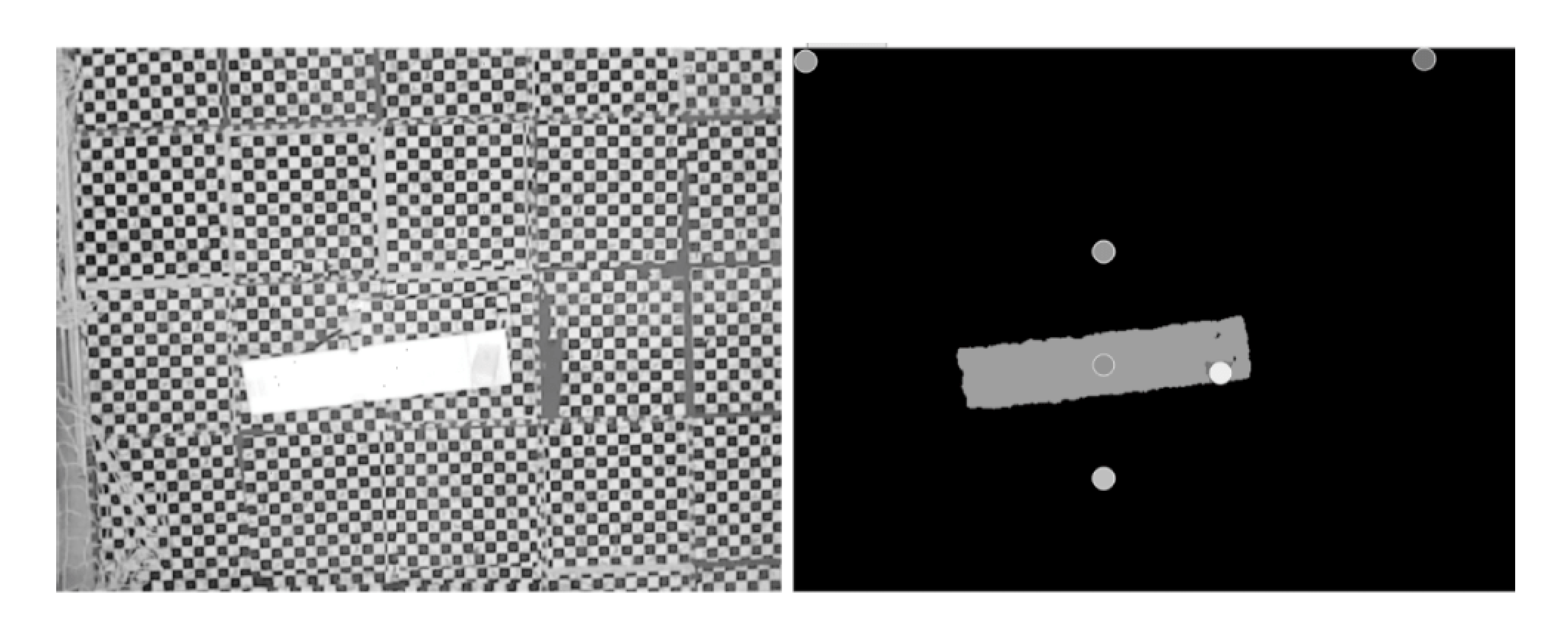


图 7.4.11　程序中所识别出的窗户位置

2. 手势控制

此应用主要实现的功能为通过操作人手的移动来控制飞行器的前、后、左、右飞行以及起飞和锁定，图 7.4.12 所示为控制示意图。

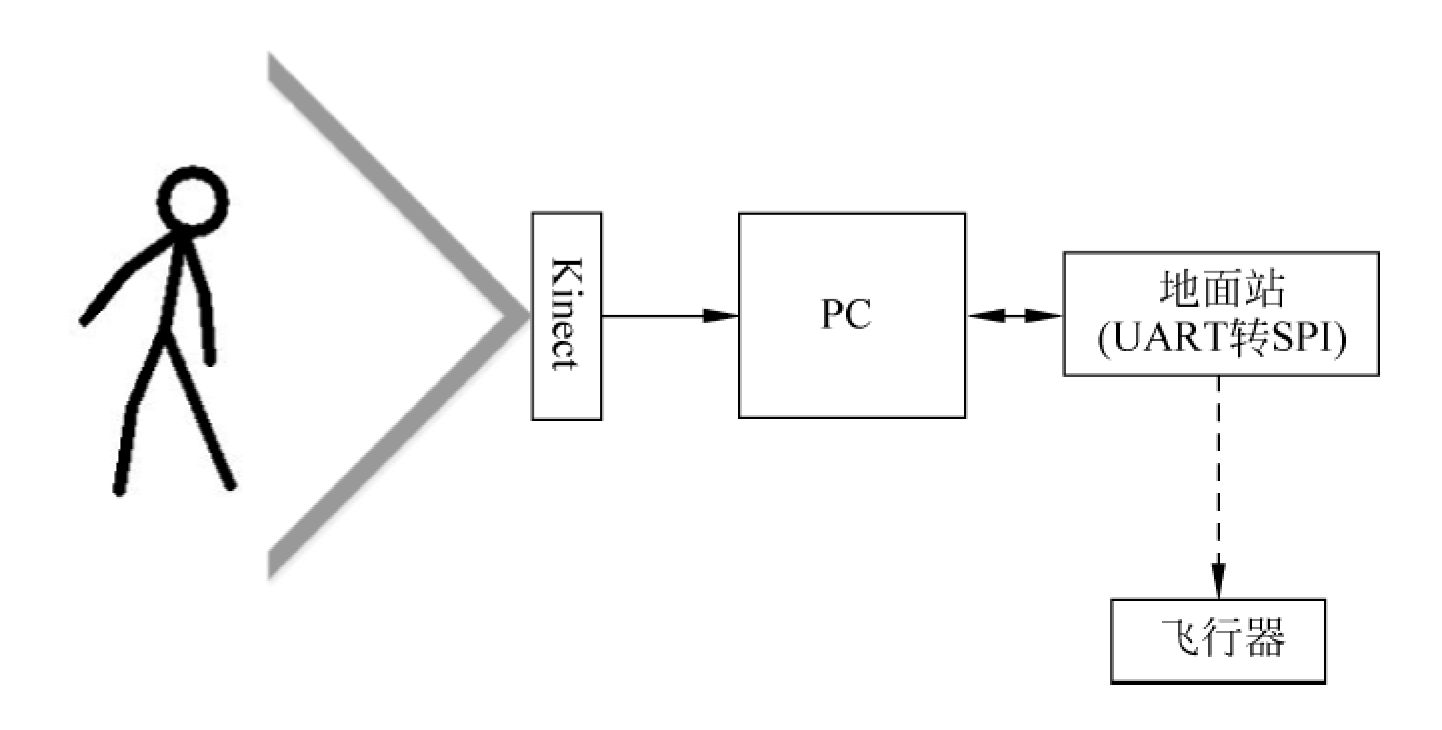


图 7.4.12　手势控制示意图

1）手势控制流程图

图 7.4.13 所示为手势控制流程图。

2）手势识别及控制

通过 Kinect 所获取的骨骼数据，可以得到人的每个关节点的坐标，这些关节点中需要用到的有头、左右手和左右肩，人摆出相应的姿势，上位机通过判断这些骨骼节点的位置来判断人的当前姿势。

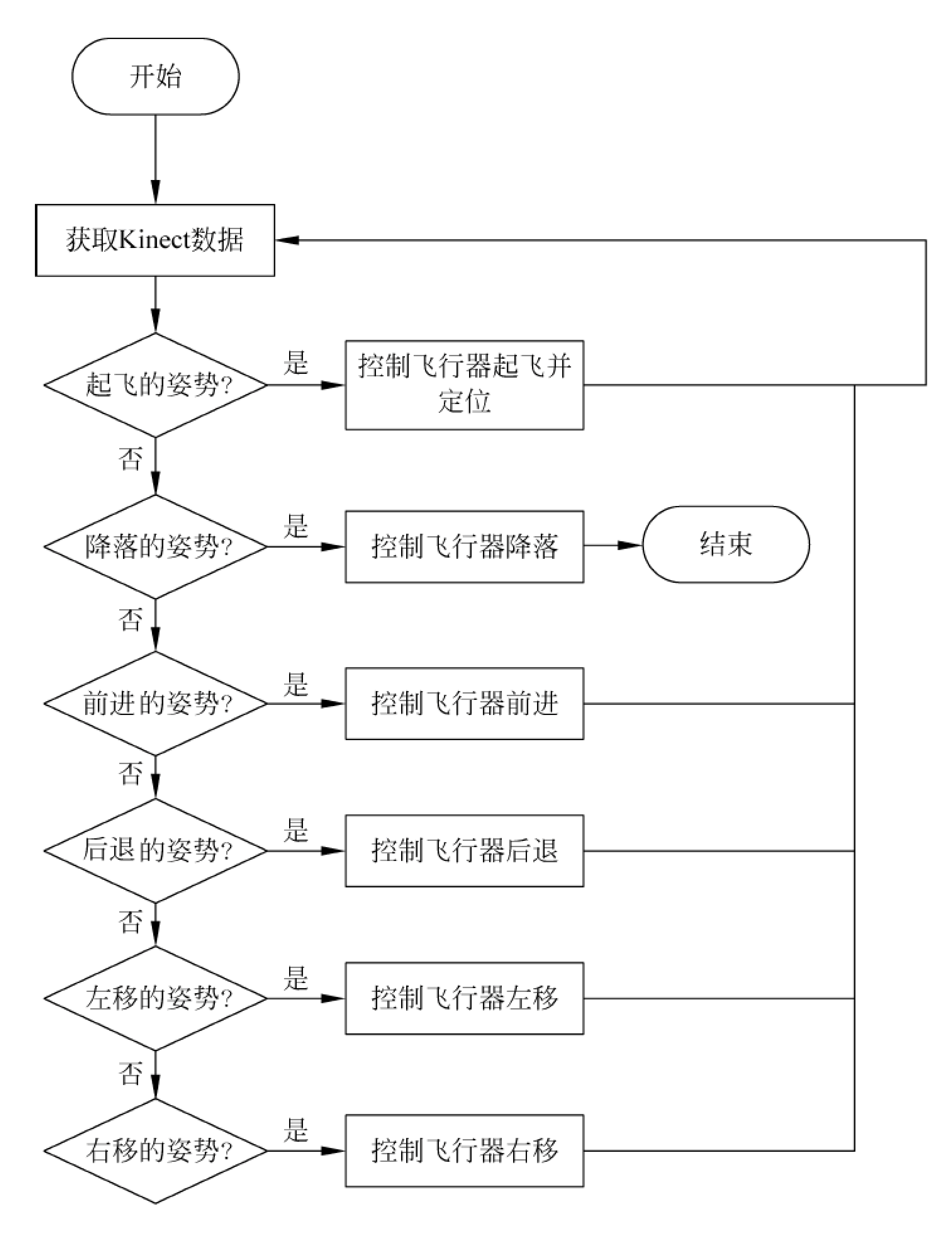


图 7.4.13 Kinect 手势控制流程图

对飞行器的控制主要分为 3 个部分，即控制其起飞、降落以及前后左右动作。通过 3 种不同类型的姿势来实现这 3 部分的控制。

3）起飞

```
startFlyGes = rightHand.Y > SPIne.Y && (rightshoulder.Z - rightHand.Z > 0.2); //起飞
```

从 Kinect 得到数据后，对右手、右肩以及脊椎的位置进行判断，只要做出右手高过脊椎并且向前伸的动作就判断为操作者想要控制四旋翼飞行器起飞，上位机就会对四旋翼飞行器发出解锁并定位起飞的命令。

控制飞行器的前后左右。对飞行器的前进、后退、左移、右移的控制是根据操作者的

右手位移来实现的：计算机会识别操作者右手的位置，将当前帧数据和上一帧比较，如果操作者的手向前移动，那么就会对飞行器发出前进的命令，同样，操作者的右手在其他几个方向上的位移也会反馈给计算机，从而通过计算机控制飞行器做出相应的移动。

降落：

```
startLand = leftHand.Y > head.Y && rightHand.Y > head.Y
```

对飞行器降落的判断是根据操作者的左右手以及头部位置来实现的，当操作者做出左右手上举过头的动作时，计算机就会判断操作者想要控制飞行器降落，于是就会对飞行器发送相应的指令。

7.4.5　总结与展望

1. 总结

本课题主要完成的任务如下。

① 通过 Kinect 获取四旋翼飞行器的移动状态以及深度数据，并对其进行分析处理。

② 通过 PC 进行图像处理，得到四旋翼飞行器以及目标物体的位置，计算四旋翼飞行器的飞行轨迹，并且对其进行控制。

③ 实现对四旋翼飞行器的实时跟踪及定位，控制飞行器飞行。

④ 控制飞行器完成寻窗、钻窗表演动作。

2. 展望

① 深度摄像头。随着机器视觉变得越来越重要，深度摄像头的出现使得能在这一领域有更大的发展。使用深度摄像头可以做许多事情。深度摄像头的主要应用有三维人脸识别技术。随着计算机技术和模式识别技术的飞速发展和三维数据采集技术不断成熟，近 20 年三维人脸识别研究受到学术界越来越多的关注，三维人脸识别已为当前模式识别技术和生物特征识别技术的研究热点。三维人脸识别的基本流程如图 7.4.14 所示。三维人脸识别具有以下几点明显优势：①三维数据中包含人脸的深度信息，从图像信息量上就更加丰富；②三维数据在三维空间内可以任意旋转变化，可以从不同的角度观察三维物体的差异；③三维人脸识别主要基于面部的深度信息，对于日常化妆的影响和附带金属物品等影响具有一定的健壮性。

② 手势提取与识别。手势识别是一种人机交互技术，以人手部的动作直接控制计算

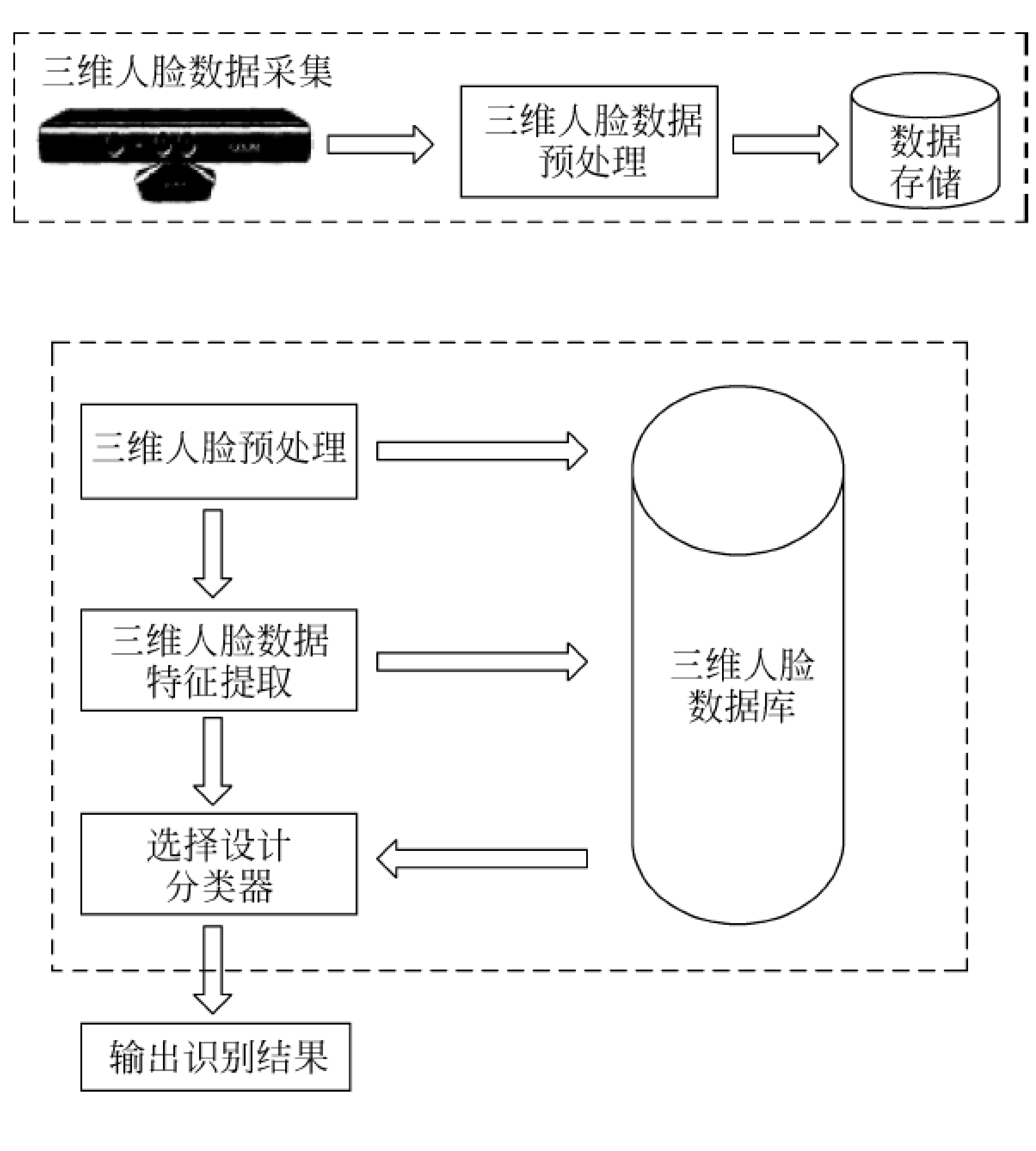


图 7.4.14 三维人脸识别基本流程

机的输入,与传统的键盘、鼠标等相比具有自然直观、易于学习等优点。使用计算机视觉技术进行手势采集后,需要对获得的图像进行手势分割,将手势从图像中分离出来。手势分割常用的方法主要有增加限制法、肤色检测法等。其中增加限制法通过佩戴有颜色的标志物或使用固定颜色的背景等对采集环境进行限制,从而通过特殊颜色进行手势分割。这种方法降低了手势的自由性,但在采集精确度上有较大优势。肤色检测法根据肤色在颜色空间分布的特点将图像转换到相应的颜色空间进行阈值分离。使用肤色检测法可以直接从图像中分离出肤色的区域,但目前的技术主要存在以下问题:容易受到复杂背景和复杂光照条件影响,不能与脸部等与肤色相近的位置重叠,不能穿戴非肤色的手套等。使用 Kinect 作为手势采集的工具,将深度图转换为三维点云,利用深度信息进行手势提取和识别分析,相比于二维手势识别有更多的优势。随着深度摄像头越来越便宜,其优势也变得越来越突出了。

③ 四旋翼飞行器。四旋翼飞行器现今有着众多的研究方向和应用领域,从国内外研究情况以及科技发展动态来看,其发展趋势主要有以下几个。

微型化。通过减少飞行器的体积和重量,减少能源消耗,从而提高飞行器的飞行时间和飞行灵活度。

智能化。随着计算机技术的迅速发展,可以提高飞行器的自适应能力和自主飞行的能力,使其能够自动执行更多更复杂的任务。

多功能性。依靠微电子技术的进步,使飞行器可以拥有更多的功能,从而有能力执行各种复杂的操作和任务,应用于更多的领域。

采用新的能源。使用高效的新能源能够减少飞行器的重量,增加飞行器的载重,从而延长飞行时间。

在这些趋势中,微型化使得四旋翼飞行器在军事和民用领域都有广阔的应用前景,根据四旋翼飞行器的发展现状和相关高新技术发展趋势,预计它将有以下发展前景。

未来的微小型四旋翼飞行器将完全能够达到美国国防预研局对 MAV(Micro Air Vehicle,微型飞行器)基本技术指标的要求。它将是一种有 4 个旋翼的可飞行传感器芯片,是一个集成多个子系统(导航与控制、动力与能源、任务与通信等子系统)的高度复杂 MEMS 系统。不但能够在空中悬停和向任意方向机动飞行,还能飞临、绕过甚至穿过目标物体。此外,它还将拥有良好的隐身功能和信息传输能力。

微小型四旋翼飞行器的编队飞行与作战应用。单个微小型飞行器的有效载荷量毕竟有限,难以有效地完成任务,而编队飞行与作战不仅可以极大地提高有效载荷量,还能够增强其突防能力。

参 考 文 献

[1] 佟刚，郝旭. 旋翼机在中国的发展现状及前景展望[J]. 中国民用航空，2014，(11)：97-98.

[2] 王连波. 浅谈无人机的发展现状及发展趋势研究[J]. 科技与企业，2013，(14)：349.

[3] 聂博文，马宏绪，王剑，等. 微小型四旋翼飞行器的研究现状与关键技术[J]. 电光与控制，2007，(06)：113-117.

[4] 赵晓颖，温立书，么彩莲. 欧拉角参数表示下姿态的二阶运动奇异性[J]. 科学技术与工程，2012，(03)：634-637.

[5] 王励扬，翟昆朋，何文涛，等. 四阶龙格库塔算法在捷联惯性导航中的应用[J]. 计算机仿真，2014，(11)：56-59.

[6] 李俊，李运堂. 四旋翼飞行器的动力学建模及 PID 控制[J]. 辽宁工程技术大学学报（自然科学版），2012，(01)：114-117.

[7] 彭丁聪. 卡尔曼滤波的基本原理及应用[J]. 软件导刊，2009，(11)：32-34.

[8] 万晓凤，康利平，余运俊，等. 互补滤波算法在四旋翼飞行器姿态解算中的应用[J]. 测控技术，2015，(02)：8-11.

[9] 蒋伟，文昱. SPI 总线及其在单片机系统中的应用[J]. 科技广场，2008，(10)：201-202.

[10] 苏建志，王冰峰. I^2C 总线及其应用[J]. 现代电子技术，2004，(22)：22-24.